An Interop Publication

CONNEXIONS®

The Interoperability Report

April 1993 Volume 7, No. 4

ConneXions — The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published monthly by Interop Company, 480 San Antonio Road, Suite 100, Mountain View, CA 94040, USA. 415-941-3399. Fax: 415-949-1779. Toll-free: 1-800-INTEROP.

E-mail: connexions@interop.com.

ISSN 0894–5926

From the Editor

April already, and I didn't manage to find a spoof article for this month! I hope you will at least appreciate the title of our first article. Gerard Parr and Piotr Bielkowicz describe a *Self-Stabilizing Bridge Protocol* designed to address weaknesses in the IEEE *Spanning Tree Algorithm,* a fundamental part of "bridge theory." Bridges and routers are fundamental building blocks for all modern computer communications networks, and we feel it is important, from time to time, to delve into the details. Your comments are appreciated as always.

The *Internet Gopher* has been described several times already in this journal. Gopher belongs to a group of innovative new applications and services which has significantly changed the way people work in the Internet. OSI's X.500 Directory Service has also been the subject of more than one *ConneXions* article. It may perhaps seem odd that two such "culturally diverse" systems could be combined, but that is exactly what has happened at the University of Minnesota (U-M). Timothy Howes describes the Gopher to X.500 gateway software which gives Gopher users access to X.500 transparently through their Gopher user agents. The Gopher to X.500 gateway service at U-M currently handles over 5000 connections per day, making it the single largest user of X.500 services at the university.

It is difficult to say anything about the *International Telecommunication Union*'s TELEDOC project without mentioning the efforts of Carl Malamud. In 1991, Carl obtained permission from the ITU to make CCITT documents available electronically via the Internet. Carl spent many months scanning and converting documents into a format that would be accessible by most Internet users. After less than a year, the ITU terminated this "experiment" and indicated that they preferred to operate electronic distribution of documents themselves. The details of how this all came about can be found in Carl's book *Exploring The Internet,* the second printing of which contains an announcement for the TELEDOC service. I like to tease Bob Shaw, the author of this month's TELEDOC article, by saying that Carl "shamed you into offering this service," but this really is a gross simplification of the state of affairs. We should not forget that the ITU has so far derived considerable revenue (which supports further standardization work) from the sale of printed standards documents. Contrast this to the "standards should be freely available to anyone attempting to implement them" attitude, and you have the perfect setup for an all-out "war" between "us" and "them." It is indeed encouraging to see the TELEDOC effort being funded and implemented by the ITU, and we look forward to providing you with more information about this project in future editions.

A Bridge Too Far

by
Gerard Parr, University of Ulster
and
Piotr Bielkowicz, London Guildhall University

Introduction

This article describes a number of improvements which were incorporated into a *Self-Stabilizing Bridge Protocol* (SSBP) [2,6,7] that was designed to address weaknesses uncovered in the IEEE 802.1 (d) Spanning Tree Algorithm [5]. The improvements include a revised bridge forwarding database tuple construction to provide additional filtering power at active bridges, facilities for N-port bridge cooperation, and operations to enable backup bridges to make more efficient use of their resources prior to load-sharing. At the time of writing, this work is currently under consideration by the IEEE 802.1 committee in the US, with a view to identifying how it relates to and complements the existing industry standard.

Loops

Over recent years, reliability of communications systems has become an overwhelming concern to network managers. In order to increase reliability at the multi-LAN scale, it is usual to install parallel bridges, thereby providing alternate paths between neighbouring LANs. The inherent problem with such an approach is that it may lead to more than one bridge forwarding the same traffic to and from the adjacent LANs which they connect. This can cause duplicate packets, unnecessary traffic loading and, in some cases infinite looping of packets which severely degrade the performance of the communications channel they traverse. The IEEE 802.1 committee has defined an algorithm to address these problems, known as the *Spanning Tree Algorithm*. Although the underlying physical topology may be an arbitrary mesh, this scheme prunes the topology to a logical topology with no cycles. It does so by electing a single *Root Bridge* for the entire system, and subsequently electing a single *Designated Bridge* (active) for each LAN, with all other parallel bridges assigned "Backup" status (passive). After the algorithm has converged, the Root is responsible for maintaining the logical tree, by periodically transmitting HELLO packets on all of its ports, which traverse the logical tree with the assistance of *Designated* bridges. On receiving such HELLO packets, the *Designated* bridges assume the logical topology is intact and forward the respective HELLO packets on their outward ports. However, if a Designated bridge does not receive an HELLO packet within the defined timeout, it must assume that the logical connectivity has been broken, and takes steps to regain status-quo.

From a detailed investigation of the Spanning Tree Algorithm, several weaknesses in its design were discovered including the fact that it does not automatically make any attempt to limit the scope of maintenance traffic to those areas with cycles, and it is based on a centralized philosophy and as such, the failure of the *Root* forces the remaining bridges to recompute their states from scratch.

BPDU

A new algorithm was designed by the authors [2] to overcome most of the problems with the existing standard. In particular, having pruned the physical topology to a logical tree by use of special "Loop Detect" *Bridge Protocol Data Units* (BPDUs), the new algorithm placed the responsibility of maintaining the logical connectivity on the *Backup* (off) bridges. The consequence of this feature was that the bridges concerned with maintaining the logical topology could fail, without any disruption to the existing connectivity, as would be the case in the IEEE standard. By use of additional facilities the new algorithm restricted the scope of maintenance traffic to those areas with cycles.

Recently, a number of improvements were incorporated into the design, relating to: *Global Bridge Cache* (GBC) tuple construction; Forwarding Database structure and use; more efficient use of non-forwarding bridge resources; restriction of Topology Reconfiguration BPDUs to loop-zones; and facilitating N-port bridges. The following sections discuss the relevant improvements to the new algorithm.

Extended Forwarding database

During the design phase, some thought was given to the structure of the forwarding database. Currently, the IEEE approach [5] is for the forwarding database to store: `<rxport#,MAC-layer-address,ttl>` tuples, where *rxport* is the port on this host on which the BPDU was received, *MAC-layer-address* is its hardware address (e.g., 48 bit in Ethernet) and *ttl* is a time-to-live quantum. The reason for storing only the MAC-layer-address is purely commercial. By storing only the MAC-layer-address, the bridge becomes more portable to a wide range of systems that adopt IEEE 802 protocols. There are no theoretical reasons why the bridge cannot store additional attributes in the tuples. Given that IEEE 802 imposes a protocol address space of either 16 or 32 bits, it would be a straightforward task for the bridge to be able to store, not only the MAC-layer-address of a station, but its protocol address as well. The obvious reason why current bridges do not get involved with anything other than the MAC-layer-address is to maintain protocol independence. This is not assumed to be a problem. The more sinister implications of not storing the protocol address attribute would become apparent during initial network operation.

Broadcast storms

For example, if a large extended-LAN is made operational after a lightening strike, the broadcasting of *Address Resolution Protocol* (ARP) ARREQs (Plummer [9]), or indeed other resource locating packets such as *rwho* under UNIX, could well occur. If the number of stations on each LAN is large (50–100), and the number of LANs is anything up to 50, one can anticipate there will be a considerable burst of broadcast traffic during this initial phase [3].

Since current bridges under IEEE only cache the MAC-layer-address, they do not have the "knowledge" on hand to prevent the forwarding of broadcast packets to destinations which have yet to "speak." Consider the operation of the bridge when dealing with ARREQs. The ARREQ arrives with source parameters and target protocol address. If the target in question exists on the same network as the source request, the IEEE bridge will not have enough knowledge on hand to prevent itself from forwarding this packet to remote LANs. What is missing is knowledge of protocol addresses. If the bridge cached protocol-layer-addresses, and MAC-layer-addresses, it could provide more filtering power. The PROXY-ARP bridges store `<rxport,MAC-layer-address,protocol-layer-address,ttl>` tuples, in their caches [10]. The decision was made to adopt the same tuple format so as to provide extra filtering capability to the bridge, without any side-effect to the bridge self-stabilizing algorithm. Indeed, with this information on hand, adoption of such a scheme complements the operation of the bridge protocol as discussed in the following sections. Further support exists for such an extended forwarding database.

It has been suggested by Currie [3] that MAC bridges which only examine MAC-layer addresses are not sufficient for controlling a large campus network, and that more sophisticated filtering is required. This implies that the bridge be capable of examining additional fields within the MAC frame. Given the benefits that could be accrued by such a policy, this scheme was incorporated into the new bridge protocol design.

continued on next page

A Bridge Too Far *(continued)*

Backup bridge behaviour

A bridge operating IEEE 802.1 d/d8 effectively blocks traffic whilst it relearns the topology after a change has taken place. The main reason for this is to observe and reflect that host locations might be accessible via different bridge ports. This might be caused by addition of a particular bridge (with better MAC-layer-address than the current *Root*), or the failure of an active bridge. This may render certain forwarding database tuples corrupt, and hence the bridge must be prudent and relearn the details. As discussed, this relearning phase can cause a blocking of traffic, because the newly elected *Forwarding* bridge has to build its forwarding database from scratch. With IEEE 802.1 d/d8 those bridges which are in *Backup,* do not process any data packets, and thus cannot build a forwarding database. When such a bridge transits from *Backup* to *Forwarding,* it has to then attempt to build its forwarding database, incurring the usual forwarding delay (e.g., 30 seconds), which effectively blocks traffic and may cause any LLC type 2 connections running through the bridge to be "torn-down" with obvious disruption to the application concerned [3, 16]. This must be frowned upon. It is viewed as an unacceptable feature, and incorporated into the new algorithm is a method for *Backup* bridges to build their forwarding databases on the "fly," that, for the most part, alleviates the problem.

Building Forwarding database tuples

Let us assume that a bridge is in *Backup* state. The IEEE approach is for that bridge to discard all data packets, and only accept certain BPDUs. Even though the bridge is logically switched off, it still has memory and processing capacity which is sitting "idle." Consider the following scenario:

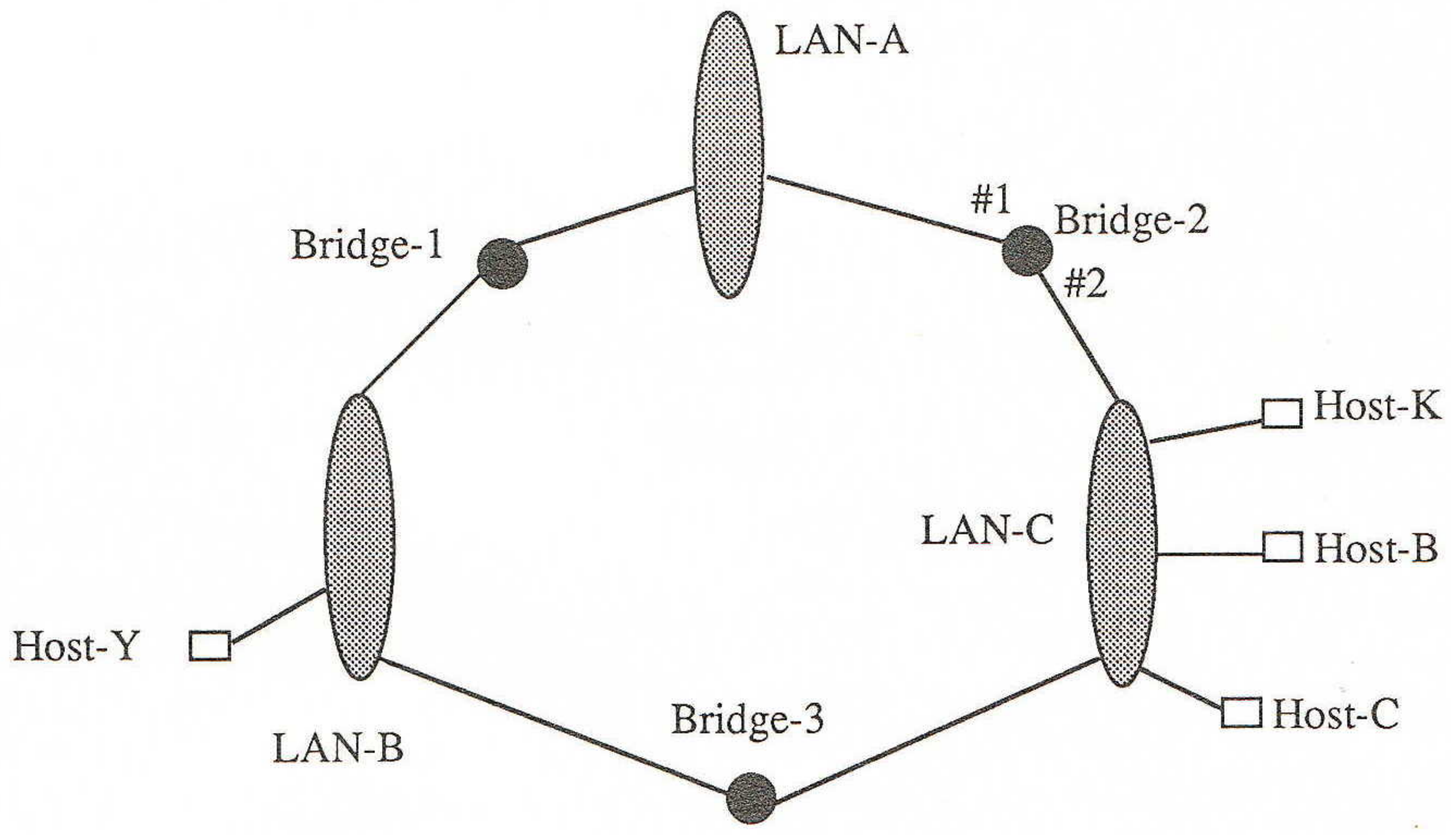

Figure 1: Building Forwarding Database Tuples on the "Fly"

Referring to Figure 1, Bridge-2 (B2) is down, but it can still "listen" to the packets that are received on each of its ports. It will process BPDUs in the usual fashion, but in addition, it breaks the rule imposed by IEEE, and processes any data packets it receives. The processing relates to the creation and management of its forwarding database tuples, without any forwarding decisions being involved. The overwhelming justification is that there is no point in having all that processing power and memory sitting idle, when it can be utilized to perform a constructive operation. Going back to the diagram, B2 can observe Hosts K, B, and C locally on its port #2. Should Host-B become more adventurous and issue a broadcast packet to Host-Y, then B2 can observe a packet from Host-B coming in on its port #1.

If B2 is establishing forwarding database tuples, it will observe an existing tuple entry for the address of Host-B. The fundamental filtering operation is based on a `<target address, rxport#>` relationship in the forwarding database. B2 could become "confused" as to the port through which Host-B is obtainable. Is it both #1, and #2? Or is it the last port# on which this address was observed, in this case, #1? The problem is highlighted whenever a bridge fails requiring B2 to transit to forwarding. Subsequently, should Host-K issue a packet to Host-B, its local neighbour, B2 should be provided with the capability of forming the correct decision (i.e., block, or filter).

Building on the "Fly"

The scheme adopted is for B2 to create forwarding database tuples on the "fly," based on observed data traffic. There should only be a single tuple for a particular host. Should a packet be received from a host which is already cached in the forwarding database with a different `rxport#` attribute, the currently received packet should be discarded, and the existing forwarding database tuple deleted. Hence, no tuples will remain for this conflicting host in the forwarding database. This is purely a prudent exercise. It is assumed that Host-B will retry communication subsequently, and B2 will observe this packet and cache it on the particular port#. Thus, when the topology contains cycles, B2 will cache packets it observes on a single port#, deleting all others that are observed on more than one port because of the loop. What is provided then, is a mechanism whereby a *Backup* bridge can collate the high "percentage" addresses of those hosts that are most active.

There is no suggestion that B2 will have all the addresses it requires for fully fledged filtering operations, but the information on hand should approximate to more than currently gained in the forwarding delay period required under the IEEE Spanning Tree Algorithm. Since B2 might have been in *Backup* for some time, it gives it the opportunity to build up a comprehensive "snapshot" of the locally connected hosts. Hence, more efficient use is made of the passive bridge processing/memory resources, paving the way to avoid unnecessary "blocking" that is currently caused in IEEE 802.1 d/d8 bridges. This scheme is incorporated into the new bridge algorithm, but it may well apply as an extension, or modification to the existing standard.

To make the algorithm sensitive to cyclic and acyclic areas of the topology, it employs a strategy whereby each bridge determines the ports through which any existing cycles can be accessed. These will then be used to construct port association records for the particular bridge in the loop, i.e., *Loop Port Associations* (LPA), that will subsequently be accessed to provide the port identifier (if any) via which the traffic should be forwarded. Obviously, if there is no LPA then the respective traffic is discarded.

Notification of topology change

The Spanning Tree Algorithm [5] provides a mechanism whereby a bridge which detects a topology change informs the *Root,* and the *Root* in turn informs all other bridges in the connected system to "relearn" their connected host addresses and locations. It employs a *Topology Notification* BPDU which informs bridges to reduce the time-to-live attribute of their current forwarding database tuples considerably (usually this is a function of the forwarding delay e.g., 30 seconds). Each bridge effectively "blocks" transmission to the majority of hosts whilst it is in this relearning stage. Such a scheme has serious consequences to the protocols running on the system, and can disrupt the performance of certain applications which adopt a connection oriented approach to communication.

continued on next page

A Bridge Too Far *(continued)*

Any host addresses "learnt" after this mark are treated as valid and are not subject to removal after the newly imposed timeout has fired.

The new algorithm employs a topology change notification, but it only circulates within the relevant Loop-Zone. Conversely, Loop-Free-Zones are not affected by the topology change, so there is nothing to be gained by forwarding notification of this phenomenon to that area of the topology. Hence a bridge which issues a *Periodic* BPDU, and does not receive same by return, will transit to *Forwarding* and inform the existing "active" bridges of this fact. Another BPDU is adopted, which has the same structure as the other BPDUs, but with an opcode = *Reconfig*. The *Reconfig* is forwarded on all ports by the newly established bridge. An existing active bridge which receives such a *Reconfig* automatically imposes a restriction on the life of its forwarding database tuples as may be deemed applicable.

Loop-free Zones

Whilst there is an obvious desire to restrict algorithm state maintenance traffic from entering Loop-Free-Zones, the question must be asked: what desire is there for having topology change *Reconfig* BPDUs entering all sectors of the extended-LAN? Since *Reconfig*s are caused by previous *Backup* bridges transiting to *Forwarding,* should not that same BPDU be restricted from entering Loop-Free Zones? To answer these questions one must consider the reaction of an active bridge when it receives a *Reconfig* BPDU. On receipt of a *Reconfig* such a bridge examines its forwarding database tuples, and all those with a time-to-live period in excess of the current forwarding delay period, will be assigned a reduced time-to-live, usually the value of the forwarding delay period. The net consequence of this action is that certain addresses will expire, and will have to be relearnt, possibly causing a delay to the respective end-users who are attempting to communicate with the host addresses concerned.

The IEEE algorithm issues topology change BPDUs to all areas of the connected system, causing all bridges to place their forwarding databases on reduced time-to-live. From this, it is obvious, that whilst the topology change would have occurred in a Loop-Zone, it will disrupt the smooth operation of the entire system, including Loop-Free Zones. After investigation, it was decided to restrict topology change announcements to Loop Zones only, as it reflects a change in link status in that area only. It may be argued that a topology change should be forwarded to all areas of the topology, purely to be prudent, but since the change will not effect the Loop-Free Zone, there is nothing to be gained. *Forwarding* bridges in the Loop-Free Zones should be permitted to continue as usual, without undue interruption.

Loop–Port associations

As discussed in [2], a cache known as the *Global Bridge Cache* (GBC) is kept at each bridge. Let us first re-state the construction of the Global Bridge Cache tuples:

```
<srcbridid,issueport,rxport,opcode,hopcount,ttl>
```

where:

`srcbridid:` 48-bit address of the bridge which issued this BPDU

`issueport:` the port on the source bridge on which this BPDU was sent

`rxport:` the port on this bridge on which this BPDU was received

`opcode:` the opcode of this BPDU

`hopcount:` the minimum distance of the source bridge from this bridge (via issueport and rxport)

`ttl:` time to live for this tuple (set by management)

Global bridge cache updating

There may be more than one tuple per "srcbridid" in the GBC. Such a collection of tuples, corresponding to BPDUs sent by the same source bridge-srcbridid, is called a *srcbridid tuple group*. When a BPDU (e.g., INIT) arrives at a bridge, the algorithm performs the following GBC update procedure:

```
if BPDUs srcbridid is not in GBC then
  add corresponding tuple to GBC
  forward BPDU on all ports <> rxport
else if (* source bridge is in GBC tuples*)
  (BPDU's issueport is not in srcbridid tuple group)
  and (BPDU's rxport is not in srcbridid tuple group) then
  add corresponding tuple to a tuple group of srcbridid bridge
  forward BPDU on all ports <> rxport
else
  drop the BPDU
```

So, a new tuple is added to the GBC and a BPDU is forwarded when there is no record of the source bridge of this BPDU, or when this BPDU is not the first one received from the source bridge, but its `issueport` and `rxport` are different from corresponding ports of "earlier" BPDUs.

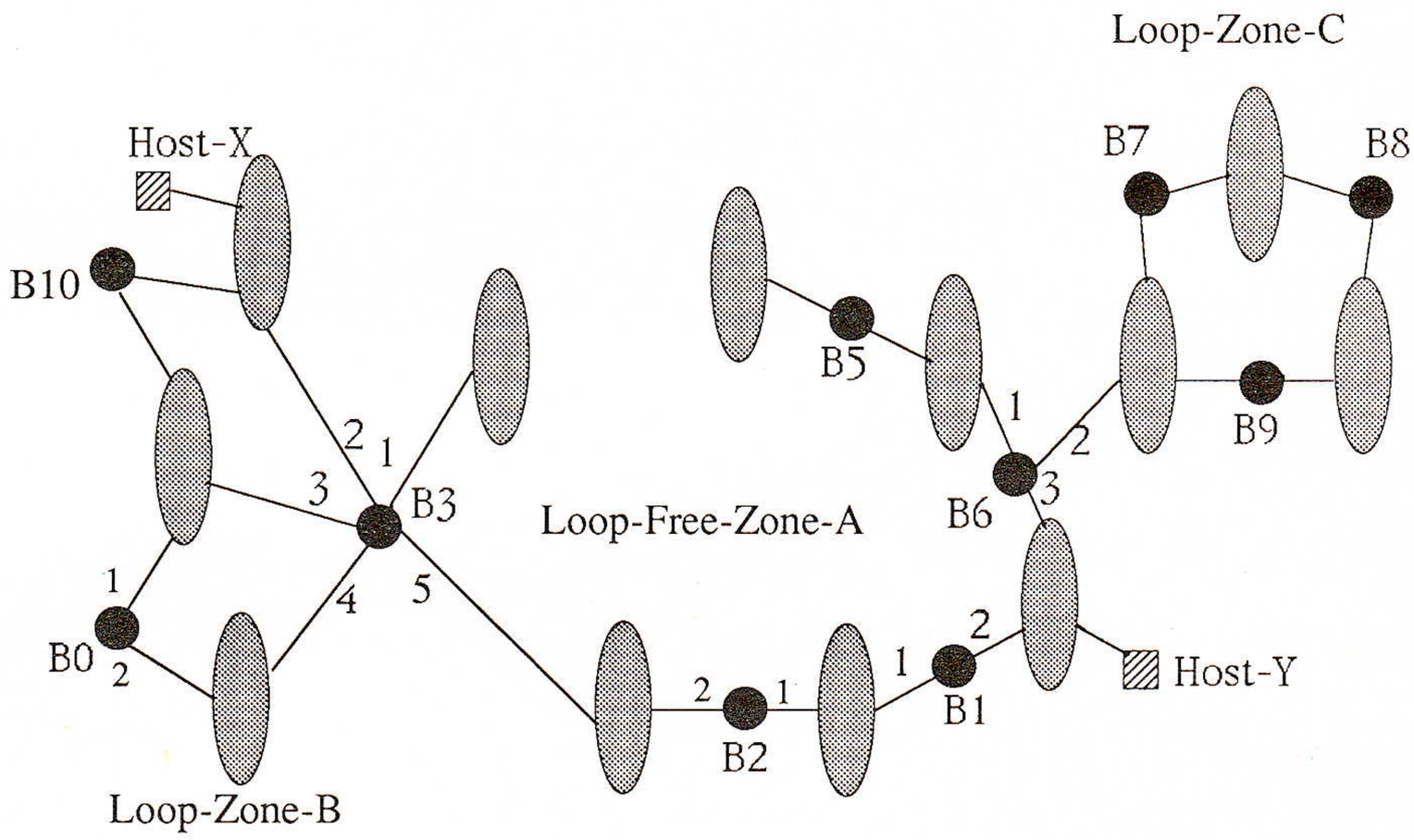

Figure 2: N-Port bridge configuration with 2 loop-zones

Referring to figure 2, let us assume that all bridges are activated approximately at the same time and bridge B1 issues an INIT BPDU on each of its ports i.e., 1 and 2 as usual. These INIT BPDUs propagate toward B2 and B6 respectively. Eventually, the B1 INIT arrives at B3 on its port 5. Bridge B3 makes a corresponding GBC tuple entry to reflect this fact:

`< B1, 1, 5, INIT, 1, xunits>` (where `xunits` is some sensible value of time). Bridge B3 now forwards the B1 INIT on all of its `ports<>rxport(5)`.

continued on next page

A Bridge Too Far *(continued)*

The INIT loops around Loop-Zone-B and arrives back at B3. Since this BPDU has both `srcbridid` and `issueport` in the GBC of B3, then the BPDU is dropped by B3. Let us notice that if this B1 INIT were forwarded then it would eventually arrive at B2's port 2. Since B2 already received B1's INIT on its port 1, it would get the (wrong!) impression that it belongs to a loop-zone! As a result of this, Loop-Free-Zone-A would be affected by maintenance traffic. Let us consider now three loop-zones (see Figure 3):

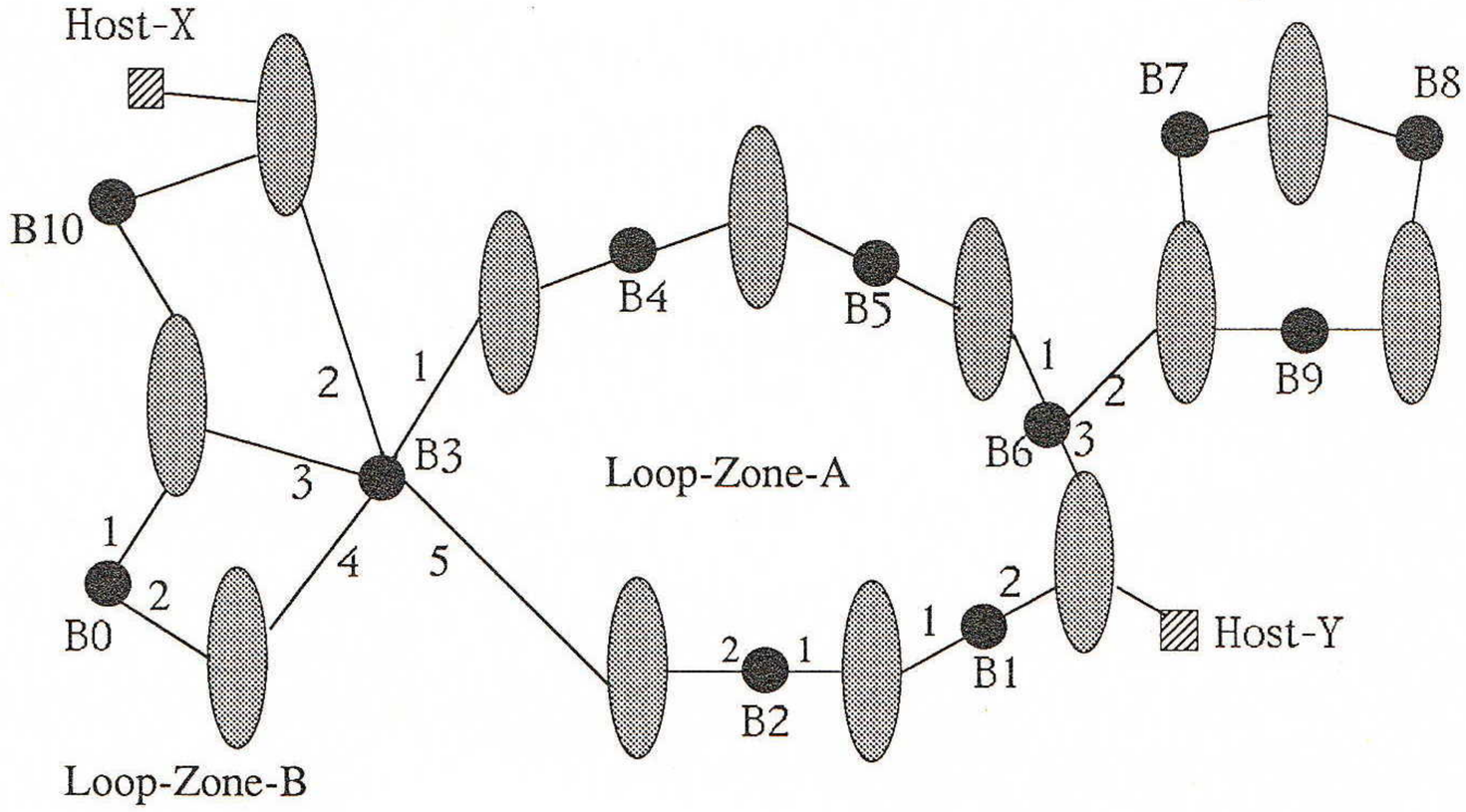

Figure 3: N-port bridge configuration with 3 Loop-zones

Let us assume again that bridge B1 issues an INIT BPDU on each of its ports. Suppose that the B1-INIT incurred delays over the B6–B4 path. When B3 receives the B1 INIT on its port 5, a GBC tuple is created (as above) and B3 forwards the BPDU. After a while the B1 INIT arrives at B3 via port 1. B3 compares the `issueport` and `rxport` values between the INIT received and the respective GBC tuple, and it discovers the difference. Thus B3 adds the new tuple to the GBC (i.e., to the B1 tuple group). The B1 tuple group looks as follows (at that point in time):

```
<B1, 1, 5, INIT, 1, xunits>
<B1, 2, 1, INIT, 3, yunits> (where yunits>xunits)
```

Next, B3 forwards the B1 INIT. On receipt of an INIT (within the required timeout) with a different `issueport` than the port on which it was received, B1 will transit to *Backup*.

Loop member detection

GBC tuples were introduced to solve the problem of packet proliferation, to detect bridge re-incarnation and to determine loop-free areas within the overall topology [2]. For 2-port bridges, on determining that a bridge is not in a loop-zone, it is then possible to flag this situation, and provide a facility for the bridge to drop all maintenance traffic it receives, thereby restricting propagation of the said traffic to the loop-zones, and reducing wasteful utilization of network/bridge resources that exist outside the loop-zones.

Backup port behaviour

We have stipulated previously that backup bridges should make more efficient use of their resources by using their *Backup* ports to build their forwarding databases on the "fly." For N-port bridges (N>2), with some ports in *Forwarding* and some in *Backup* state, the problem must be addressed carefully. Let us consider the following scenario shown in Figure 3.

The Host-X boots in Loop-Zone-B, runs ARP[9], and issues an ARREQ for Host-Y in Loop-Zone-A. Bridge B3 must be in a position to forward this ARREQ. This will only happen if no tuple exists for Host-X in the ARP-cache of B3 (as some implementations have them based on RFC 925 [10]), and it arrives on a forwarding link. Initially in B3, the backup port, 2, will observe the ARREQ and can determine that no tuple exists for this host in the forwarding database of B3, so it creates a tuple on the "fly." Since it came in on a backup port, the code will not process the packet any further. Subsequently, port 3 of B3 will receive the same ARREQ, but this time it will discover that an existing tuple is stored for Host-X, placed there by port 2 (the backup port). To provide for prudence, B3 can override any forwarding database tuples created by backup ports when it receives the same packet on a forwarding port. After the overwrite, B3 then forwards the ARREQ as required. This feature of building on the "fly" in backup ports is still employed, but in this case the backup ports concentrate on collating intra-network traffic on Host-X's LAN, and will create forwarding database tuples for possible use later when the port may be required to transit to forwarding.

Transit to forwarding

If we continue with the same B3 scenario, its port 2 may have to transit to forwarding. By use of the LPA records as mentioned previously, B3 will know that the port associated with port 2 is port 3. From this, B3 will configure and issue *Reconfig* BPDUs via port 3, thereby restricting the propagation of reconfiguration traffic to the respective loop-zone (i.e., Loop-Zone-B).

It is important to realize that bridges which are in *Backup* state will have set their Loop Member Flags accordingly. Referring to Figure 4, should the logical topology have been set, with bridge B3 *Forwarding*, let's assume B3 fails.

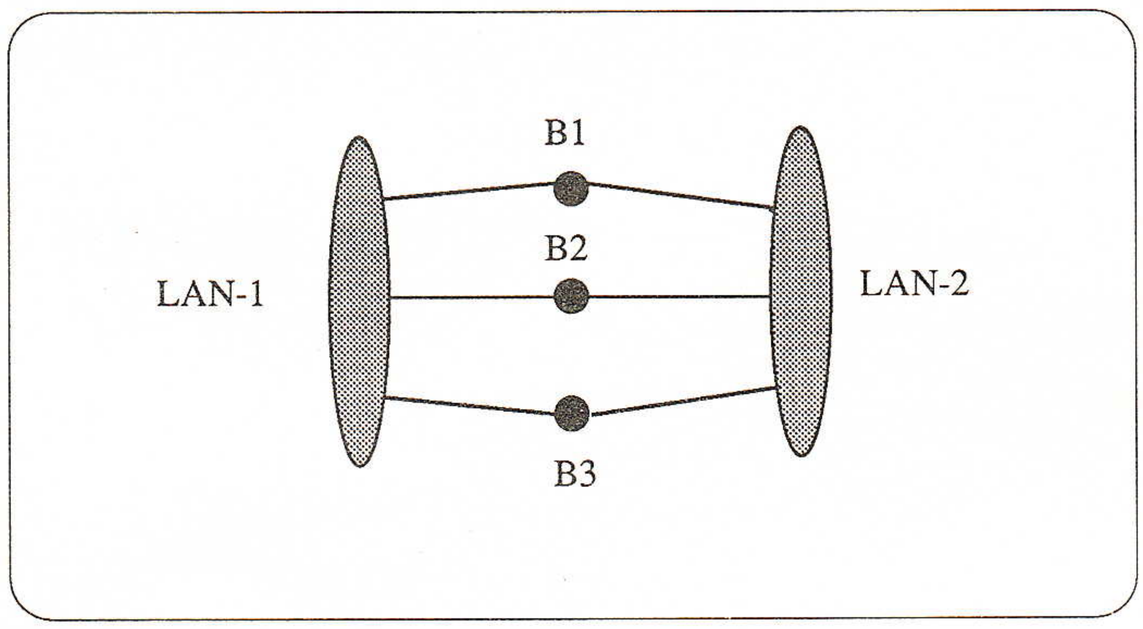

Figure 4: *Backup* to *Forwarding* Transition

It must be noted, that a bridge in *Backup* will transit to a temporary state of *Backuptest* where it issues its *Periodic* and awaits the outcome. It will subsequently transit to *Backup* again on return to its *Periodic* state or transit to *Forwarding* if the timeout fires, from this *Backuptest* state. Concentrating on the use of LPAs, the problem could have arisen that B2 would not forward the *Periodic* from B1 if B2 did not set its LPA. Since all *Backup* bridges set their LPAs just after they transit from INIT to *Backup*, this is not a problem. B2 will have its LPAs set, as indeed will B1. From this, B2 will forward the *Periodic* it receives from B1 forcing it to transit to *Backup*. By the same token, if we still assume the bridge *Backuptest* states are approximately synchronized, B1 will not forward the *Periodic* from B2, and hence B2 will transit from *Backup* to *Forwarding*.

continued on next page

A Bridge Too Far *(continued)*

On the other hand, if there is no approximate synchronization between bridge *Backuptest* states, it could well come to pass that B1 is the first bridge to issue a *Periodic* BPDU. Since B3 has crashed, and B2 is in *Backup* state, B1 will not receive its *Periodic* by return, and it will transit to *Forwarding*. When B2 subsequently issues its *Periodic* B1 will forward it, and B2 will return to *Backup*.

General traffic metrics

A detailed simulation model was designed to represent various complex multi-LAN topologies, and ran with the bridge operating the 802.1d and the new algorithm respectively. The results indicated that the new algorithm was more efficient than 802.1d as the number of Loop-Free Zones increased above a certain ratio. Consider the following scenario for a 2-port bridge.

Future issues

With the ever increasing awareness of *Integrated Services Digital Networks* (ISDN), the view now is to provide internetworking boxes that can cope with the multitude of applications running over an ISDN and SMDS infrastructures [14,15]. We can imagine IP running over such an infrastructure supporting user application software from the Multi-LAN site to the workstation in the home. The function of Remote Bridging support will still be as important, if not more so [15]. In particular, the efficient operation of any SSBP will have great impact on the level/quality of service obtainable by applications traversing the Multi-LAN to the "outside world," via, for example, SMDS [16]. Indeed, taking the application area outside our world to the likes of the proposed NASA Space Station, the requirements for fault tolerance and reliable performance are more pronounced. The quest is on for "future-proof" bridging protocols that will support, not only Narrowband-ISDN, but also the high speed switching necessary to carry Broadband-ISDN applications that require rate adaptable and high Quality of Service parameters.

Conclusion

The problems that Perlman & Varghese highlighted in [8] have been addressed, and individual solutions proposed to provide a new algorithm [2,6,7]. In addition, the facility to confine algorithm state maintenance traffic within its own cycle serves to make the overall algorithm more powerful and efficient than the loop detect discussed by Perlman & Varghese. If we have a topology which has a few "loop" bridges and many "loop-free" bridges, our algorithm would appear more sympathetic to the physical topology which exists. In addition, use of a new forwarding database design, coupled with more efficient use of *Backup* bridge facilities to create their forwarding database in readiness, leads to a more efficient design. Aside from bridge self-stabilization, the following facilities are provided:

- An extended forwarding database giving more filtering power.
- More efficient use of the resources contained in *Backup* bridges to construct their forwarding database on the "fly," and make ready for transition to *Forwarding,* and immediate operation.
- Restriction of state maintenance *Periodic* BPDUs to loop-zones only.
- Restriction of topology change (*Reconfig*s) to loop-zones only.
- A decentralized design, providing less interruption of service to the end-user.

Given the facilities presented in the previous sections, this new approach is considered to provide objectives and primitives that should be incorporated into future releases of the industry standard.

As such, it is not intended to promote this algorithm as a replacement for the existing standard, but it shoud be viewed as a source of ideas and functions to encourage debate and discussion between interested parties in the Internet Community.

References

[1] Ball, E., Tasker, R., Linge, N., Kummer, P., "A Bridge Protocol for Creating A Spanning Tree Topology Within an IEEE 802 Extended-LAN Environment," *Computer Networks and ISDN Systems,* Volume 13, No. 4–5, 1987, pp 323–332.

[2] Bielkowicz, Piotr, Parr, Gerard, "A Loop-Detect Packet Based Self-Stabilizing Bridge Protocol for Extended-LANs," ACM SIGCOMM *Computer Communications Review,* Volume 19, No. 5, October 1989, pp 72–81.

[3] Currie, Scott, Wilson, Sam, "Large Campus Networks: To Bridge or to Route?" Networking Planning Project, Edinburgh University Computing Service, 12th January 1990,

[4] Kummer P., Tasker, R., et al, "A Protocol-less Scheme for Bridging Between IEEE 802 Local Area Networks," *Computer Networks and ISDN Systems,* No. 12, No. 2, 1987, pp 81–87.

[5] IEEE, "MAC Bridges: The Spanning Tree Algorithm and Protocol Standard P 802.1 d/D," Draft Document, March 1989, pp 58–142.

[6] Parr, Gerard P., "A New Self-Stabilizing Bridge Protocol for Extended Local Area Networks," PhD Thesis, Department of Computer Science, University of Ulster, March 1990.

[7] Parr, Gerard P., Bielkowicz, Piotr, "A New Self-Stabilizing Bridge Protocol for Extended-LANs," IEEE TENCON '90 Conference on Computer and Communications Systems, 24–27 September 1990, Hong Kong.

[8] Perlman, Radia, Varghese, George, "Pitfalls in the Design of Distributed Routing Algorithms," ACM SIGCOMM Symposium, Volume 18, No. 4, August 1988, pp 45–54.

[9] Plummer, David C., "An Ethernet Address Resolution Protocol," RFC 826, November 1982.

[10] Postel, Jon, "Multi-LAN Address Resolution," RFC 925, October 1984.

[11] Salwen, H., Chiappa, J. Noel et. al., "Examination of the Applicability of Routers and Bridging Techniques," *IEEE Network,* Volume 2, No. 1, January 1988, pp 77–80.

[12] Sincoskie, W. David, Cotton, Charles J., "Extended Bridge Algorithms for Large Networks," *IEEE Network,* Volume 2, No. 1, January 1988, pp 1623.

[13] Zhang, Lixia, "Comparison of Two Bridge Routing Approaches," *IEEE Network,* Volume 2, No. 1, January 1988, pp 44–48.

[14] Horn, D. N., "An ISDN Multimedia Conference Bridge," Proceedings of IEEE TENCON '90 Conference on Computer and Communications Systems, Sept. 1990, Hong Kong.

[15] Clapp, G. H., "LAN Interconnection Across SMDS," *IEEE Network,* September 1991.

[16] Achter, D. Van, "Routing approaches for remote LAN bridging over SMDS using the spanning tree," First International Symposium on Interworking, Bern, Switzerland, November 18–20, 1992, Session 5, pp 1–15.

GERARD PARR holds a Bsc [Hons] in Computer Science and a PhD in Communications and Networks from the University of Ulster, Northern Ireland. During his PhD he obtained a scholarship to work at USC-ISI with Jon Postel. Since 1989 he has been a member of faculty staff at the Department of Applied Computing, Magee College in Derry City and has worked on a variety of projects including: Generic ATM Switch Fabric Analysis, Application of AI to Network Management, Distributed Operating Systems for B-ISDN environments. He has also participated in EC programmes including STAR, RACE, COMETT and IMPACT and SERC projects. He is a member of BCS, IEEE and sits on the Technical Editorial Board of *IEEE Network* in the USA. He is a reviewer for *Telecommunications Systems Journal, JHSN* and *Simulation* in the USA. E-mail: `cfad33@ucvax.ulster.ac.uk`

PIOTR (PETER) BIELKOWICZ is a Senior Lecturer in Computing and Information Systems at the London Guildhall University, having previously been a Lecturer in the Department of Computing Science at the University of Ulster (Coleraine). Before that he spent over ten years at the Institute of Mathematical Machines in Warsaw, first working as a Research Assistant in the Real Time Systems group where he was a Research Fellow. He also worked at Concordia University, Montreal as a Visiting Research Associate. He has an MSc from Warsaw University. His main interests include Distributed Systems and Internetworking, software engineering methodologies and simulation techniques.

The Gopher to X.500 Gateway

by Timothy A. Howes, University of Michigan

Abstract

This article describes the Gopher to X.500 gateway software developed at the University of Michigan (U-M). *Gopher* is a simple distributed document search and retrieval protocol for the Internet. X.500 is the OSI standard for distributed directory service. The Gopher to X.500 gateway software gives Gopher users access to X.500 transparently through their Gopher user agents. The Gopher to X.500 gateway service at U-M currently handles over 5000 connections per day, making it the single largest user of X.500 services at the university.

Introduction

The Gopher and X.500 protocols are very different. Gopher was developed at the University of Minnesota and originally described by a short six page paper [1]. Simplicity is one of Gopher's key objectives. It is basically a navigation and retrieval protocol, imposing very little structure on the data ultimately retrieved or the namespace in which it lives. The Gopher namespace forms a general graph structure, though in practice this is often restricted to a hierarchy. No facilities are provided in the protocol for modification or replication of data. X.500, on the other hand, is quite complicated by any standard. Developed by an ISO and CCITT committee, it is described by an International Standard [3] several hundred pages long. X.500 is a directory service protocol meant to provide navigation, naming, data storage and retrieval, and authentication capabilities. It defines a strict information framework that requires data to be well-structured and strongly typed. The X.500 namespace is hierarchical, reflecting the political and geographical boundaries of countries, states, localities, organizations, etc. Aliases can be used to provide a more general graph structure.

Despite their differences, tying Gopher and X.500 together has proved to be surprisingly easy. This article describes *Go500* and *Go500gw*, two Gopher to X.500 gateways developed at the University of Michigan. Since the gateways were deployed and announced at a Gopher developers conference, combined usage has steadily increased to over 5000 connections per day, making them the biggest users of the University of Michigan X.500 service. In addition, a number of other sites are now running gateways of their own. The first two sections give brief overviews of Gopher and X.500, respectively. The next sections describe the gateways themselves, followed by a few comments about our implementation experiences, and a look at the gateway usage in more detail. Finally, we consider possible future work in this area, followed by information on how to get the gateway software, which is freely available on the Internet.

Overview of Gopher

The Gopher protocol is based on a client–server model in which Gopher servers hold documents which may be accessed by users running Gopher clients. Only navigation and retrieval facilities are provided by Gopher. The protocol makes no provision for the modification of data. The model resembles a hierarchical filesystem, with both files (documents) and directories (menus). Menus contain lists of documents and/or other menus. Gopher documents can contain text, sound, pictures, etc. Other types of menu entries are also possible, e.g., index search servers. Each item listed in a Gopher menu has a type (document, menu, etc.), a name which is usually displayed to the user, a *selector string* used internally by the client and server to uniquely identify the item, and the IP address and TCP port number of the Gopher server holding the item. These pieces of information are tab-separated when presented to a client.

A client retrieves an initial list of menu items by connecting to a Gopher server and sending it a null line. The Gopher server responds with the list of items in its "root" menu. Menu items are separated by newlines. The entire list is terminated by a single period on a line by itself. To retrieve one of the items listed in the menu, the client connects to the Gopher server at the specified IP address and TCP port number and sends it the selector string. The server responds with the item requested, be it a document or another list of menu items.

Another type of entry that can occur is called an *index search server.* The entry for such a server is similar to those for other Gopher items, except that to access the search server a Gopher client is expected to present the selector string and a list of words for which to search, presumably retrieved from the user. The index server then returns a menu of Gopher entries that match the search criteria. The index search server was designed to allow full text searching of the various documents held by a Gopher server, but as we shall see later it can be put to other uses.

Simplicity and power

By keeping their protocol very simple and building most of the intelligence into the client software, the designers of Gopher have come up with a system that is powerful, yet simple to use and implement. Cost of entry into the Gopher world is very low. It consumes few resources, the implementations are easy to bring up and administer, and little effort is required to understand the entire system. Furthermore, it uses technology that is mature and well-understood, making Gopher software development easier. Virtually anyone with IP connectivity can run a Gopher server and offer the data they hold to the rest of the world. Namespace and data management are user-driven in this sense. This is both good and bad. It's good because it allows users to have control over their data and the ability to design an information service tailored to their needs. They don't have to seek permission from anybody to start providing a useful service. It's bad because it does not always lead to a sensible organization of data and can tend to produce a tangled namespace, making it hard to find things.

Overview of X.500

X.500 is the OSI standard for directory service, specified jointly by the *International Organization for Standardization* (ISO) and the *Consultative Committee for International Telephony and Telegraphy* (CCITT). It specifies a general distributed directory service that assumes that read operations are far more frequent than writes and that temporary inconsistencies among data are acceptable. The model is client–server based, like Gopher, but more complex. In X.500, clients are called *Directory User Agents* (DUA) and servers are called *Directory System Agents* (DSA). A DUA connects to a "close" DSA and sends its query. The DSA can either answer the query from its own data, forward the query to another DSA on behalf of the client (this process is called *chaining*), or return the address of the other DSA so the DUA can ask it itself (this process is called *referral*). Regardless of which DSA a DUA contacts, it sees the same view of the X.500 data. DUA to DSA communication is accomplished using the *Directory Access Protocol* (DAP). DSA to DSA communication is over the *Directory System Protocol* (DSP). Both protocols are defined in terms of the OSI Remote Operations Service [4].

The data itself is composed of *entries* which are organized into a hierarchy called the *Directory Information Tree* (DIT). At the top of the tree are entries for countries and international organizations. Below each country the namespace is decided by the country itself, with some guidance from the X.500 standard.

continued on next page

The Gopher to X.500 Gateway *(continued)*

In the US, for example, the *North American Directory Forum,* a group of public directory service providers, is defining a namespace that follows the existing civil naming infrastructure [5].

Attributes

Each entry in the DIT is composed of a number of *attributes.* Each attribute has a *type* and one or more *values.* The form the values can take is determined by the attribute's *syntax.* An example attribute used for holding a person's name might have type *commonName,* syntax *CaseIgnoreString* (meaning the value is a string, the case of which is ignored for comparison purposes), and the two values "Timothy A Howes" and "Tim Howes." There is little restriction on the range of syntaxes and attributes that can be defined. Two of the more interesting ones defined in the Internet X.500 pilot are *jpegPhoto* and *audio,* which are for holding pictures and sound, respectively.

Names

Entries are named by one or more of these attribute value pairs, the collection of which is termed the entry's *Relative Distinguished Name* (RDN), and must be unique among all sibling entries. The globally unique *Distinguished Name* (DN) of an entry is formed by concatenating the path of RDNs from the root of the DIT to the entry. For example, the entry for the United States is named {`country=US`}. The entry for the University of Michigan is named {`country=US, organization=University of Michigan`}.

X.500 defines operations that allow a DUA to search and browse the DIT, retrieve information from particular entries, and even modify, add and delete entries. For our purposes, only the read-like operations are of interest, search in particular. Searches can be of a single entry, an entry's children, or an entire subtree of the DIT (even if the subtree is split across multiple DSAs). Search filters are based on Boolean combinations of attributes which satisfy certain conditions, such as equality, approximate equality, substring matching, etc. So, for example a search could be made for entries with a surname equal to "Howes" or a commonName approximately equal to "Tim Howes." For a more complete treatment of X.500, see [6].

The gateways

The first pass we made at a Gopher to X.500 gateway is known as *Go500.* It is tailored to white pages usage and only gives Gopher users access to a preselected portion of the X.500 DIT. It allows subtree searching of this fixed portion of the namespace, but no browsing. It appears as an index search server to a Gopher client, which prompts the user to enter keywords for which to search. Go500 takes the keywords, forms an X.500 search filter, and performs a subtree search of the portion of the DIT for which it is configured. A list of entries matching the search criteria is returned to the Gopher client, each entry represented as a text document. When the user chooses one of these documents, the corresponding entry's attributes are retrieved, converted to text form, and displayed to the user. Various white pages attributes are displayed.

This gateway worked well to solve our initial problem, which was to provide a faculty, staff and student phone directory through Gopher without duplicating the information we already held in our X.500 database. Go500 is not a general gateway, though, and gives Gopher users no way to access X.500 databases at other organizations, nor does it allow browsing of the X.500 namespace, something Gopher users are fond of doing.

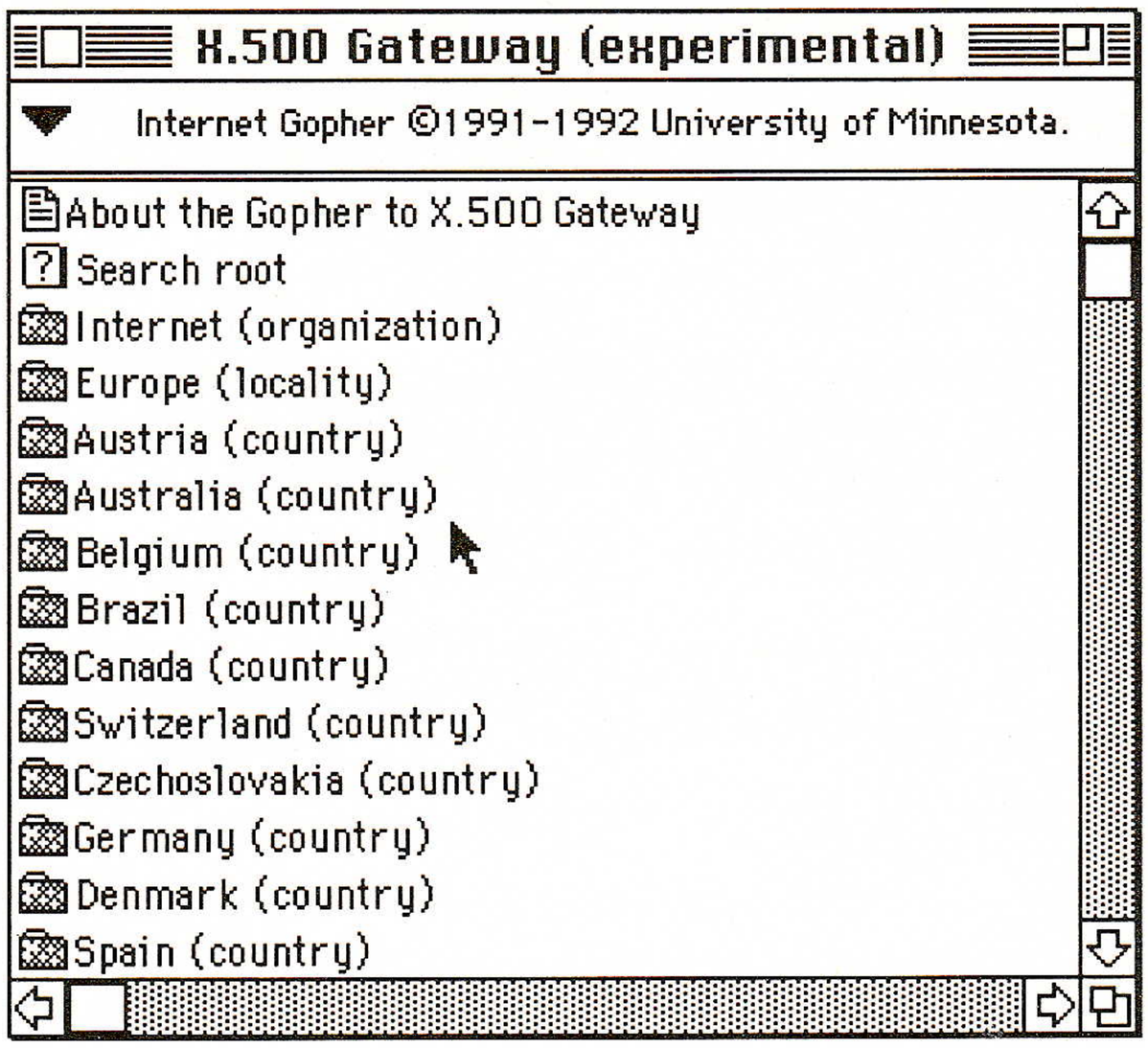

Figure 1: Root menu returned from the gateway

Go500gw

To solve this problem and provide more general X.500 access, we developed *Go500gw*. Go500gw appears to the unsuspecting Gopher client as just another Gopher server. The initial Gopher "menu" that is exported to the client consists of the list of X.500 entries at the root of the DIT. Leaf entries in X.500 appear as text documents in Gopher. Nonleaf entries appear as other Gopher menu servers (though they in fact point to the single Go500gw server, but with a different selector string). Figure 1 shows a portion of an example root menu returned by the gateway, as displayed by a popular Macintosh Gopher client. If a Gopher user selects one of the leaf, or document objects, Go500gw retrieves the contents of the entry, converts it to text form and sends it back to the Gopher client. If a Gopher user selects one of the non-leaf, or menu objects, Go500gw retrieves a list of the entry's children and sends it back to the Gopher client. In this list, nonleaf children are presented as Gopher menus and leaf children as documents, just as for the initial list. In both cases the selector string is the text-encoded form of the entry's Distinguished Name, prefixed with a one character flag indicating whether a "list" or "read" operation is required. The "list" operation is used for menus, "read" for documents. Using this simple scheme, a user can descend to any point in the X.500 DIT and view the contents of any leaf entry.

To avoid clutter in the namespace, entries for DSAs are excluded from lists sent back to the client. Also, certain attributes not likely to be of interest to the user are not displayed (e.g., those dealing with X.500 knowledge references and access control). To help users find their way, each entry is identified by its primary object class in the directory when it is displayed, for example *person* or *organization*. This information is simply included in the user-visible name of the object. This way, users can easily tell organizations from states or localities, and people from mailing lists or application processes.

Search procedure

At each level Go500gw also exports two special entries to the Gopher client. One is labelled `Read <name> entry`, where `<name>` is the X.500 Relative Distinguished Name of the entry marking the current position in the DIT. (Since there is no real "root" entry in X.500, this item does not appear in the top level menu.)

continued on next page

The Gopher to X.500 Gateway *(continued)*

This allows Gopher users to retrieve the attributes of nonleaf entries as well as leaf entries. The second is labelled `Search <name>`, and allows Gopher users to be more selective in their browsing by using the X.500 search facilities. This entry appears as a Gopher index search server. Go500gw takes what the user types, forms and executes an X.500 search query, and returns the list of results to the Gopher client in the same form as described above (menus for nonleaf entries, text documents for leaf entries). The exact form of the query applied to X.500 depends on the user's current position in the DIT and on what the user types. The gateway assumes that searches initiated higher up in the tree are looking for countries, states, localities or organizations. In this case, a one level search is performed, with a filter appropriate for finding such objects. Searches initiated lower down in the tree (e.g., below an organization entry) are assumed to be for other objects, like people. In this case, a subtree search is called for, with a slightly different filter, more suitable for finding people or other objects. Figure 2 shows how an example leaf entry is displayed.

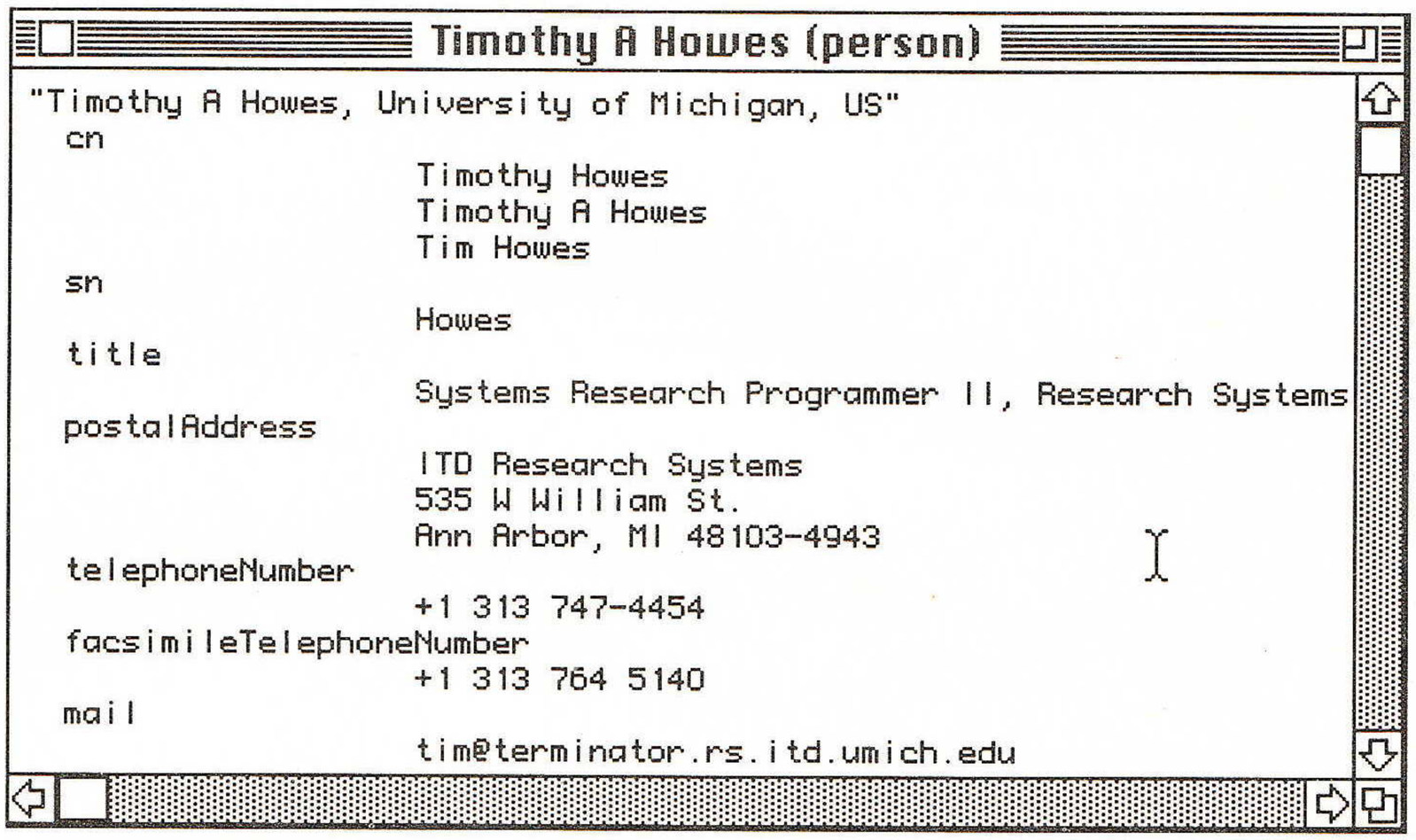

Figure 2: An example X.500 entry as displayed by the gateway

An early version of the gateway allowed users to specify their choice of one level or subtree search and required them to specify the X.500 search filter directly. Experience showed this to be more confusing than helpful to people, despite the added flexibility. In practice, the searching assumptions described above seem to work most of the time, without requiring Gopher users to know anything about X.500.

Implementation experience

Implementation of the gateways was surprisingly easy. The initial version of each required less than a day's programming time and well under 1000 lines of C code. Credit for this happy surprise belongs equally to Gopher itself, which is an extremely simple and easy to work with protocol, and the *Lightweight Directory Access Protocol* (LDAP), which is what the gateway uses to talk to X.500. Gopher is simple enough that a reasonably complete understanding of the protocol can be gained in less than an hour by reading the Gopher paper and using *Telnet* to poke at a Gopher server or two. Gopher is also very flexible, despite its simplicity. A surprising amount can be accomplished through clever use of the selector string.

The other half of the credit goes to LDAP, which provides access to most X.500 capabilities directly over TCP. LDAP uses simplified string encodings for many protocol elements, greatly relieving the encoding and decoding burden clients must bear. The University of Michigan implementation of LDAP includes a client library and simple API making it relatively easy to develop LDAP clients (like the Gopher gateways), without the usual baggage associated with an OSI application. LDAP is currently a proposed Internet Standard [2, 7].

Gateway usage

The simple Gopher to X.500 gateway, Go500, has been in use at the University of Michigan for nearly a year, providing our Gopher users with access to the X.500-based faculty, staff and student directory. Initial growth was substantial but has now slowed. Go500 currently handles between 1000 and 2000 connections per day. The vast majority of these connections originate from the University of Michigan campus. The more general gateway, Go500gw, has been in operation for about six months and is listed under the University of Michigan main Gopher menu as well as at the "Mother of all Gophers" at the University of Minnesota. Its usage has grown dramatically to over 3000 connections per day, making it the single largest user of our X.500 service. Together, the two gateways send over 5000 queries a day to our X.500 server. Usage was so great, in fact, that we started another DSA solely for the purpose of handling the Go500gw traffic. A number of other sites are also now running Go500, Go500gw, or both, making it difficult to know how much usage the gateways get over all.

Future work

There are a number of small improvements to be made to the gateways, mostly in the areas of error handling, which is currently rather minimal, and search heuristics, which could use some tuning. The current gateway provides Gopher users access to X.500. The reverse function, giving X.500 users access to Gopher, would make an interesting project, though one somewhat more complicated. X.500 attribute types and object classes must be defined to accommodate the Gopher data types. Some sensible mapping from X.500 operations to Gopher operations must also be defined, keeping in mind that there are not analogous operations in many cases. Some Gopher types are not representable at all in the X.500 world, for example the *Telnet* type, which identifies a server to which the Gopher client is supposed to *Telnet* and provide interactive access for the user.

Another approach would be to register individual Gopher servers in X.500, giving access information that would be understandable by a Gopher client. Although this would not help X.500 clients access data contained in Gopher servers, it would allow Gopher clients accessing the gateway to make use of the X.500 naming and searching capabilities in locating Gopher servers. This approach would be relatively straightforward to implement.

Conclusion

Certainly based on usage alone, the Gopher to X.500 gateways have been very successful. The relative ease of implementation has been an especially nice bonus. More interesting is what the gateway's success suggests about Gopher, X.500, and their users. The lesson about Gopher users seems to be that they are curious creatures. Browsing seems to be the most common use of and way of discovering the gateways. If there is a lesson for the X.500 community it is that simplicity and ease of use and installation are big pluses when it comes to getting a protocol out there and used by the community at large. The use of the gateways clearly shows that users are interested in the data and services provided by X.500.

continued on next page

The Gopher to X.500 Gateway *(continued)*

If they have a simple, easy to use tool on their desktop with which to access X.500, they will use it. It is only by providing such tools (and correspondingly simple configuration and administration on the server side) that X.500 will continue to grow in the Internet into a more widely-used, mature service. Gopher, on the other hand, has enjoyed, or perhaps suffered from, explosive growth over the past couple of years. Its future success depends in part on how well it evolves to handle the scaling effects it is now beginning to feel.

Availability

The University of Minnesota Gopher distribution is available for anonymous FTP from the host `boombox.micro.umn.edu`. The two Gopher to X.500 gateways described in this article are available as part of the University of Michigan LDAP distribution, available for anonymous FTP from the host `terminator.rs.itd.umich.edu`.

References

[1] Bob Alberti, Farhad Anklesaria, Paul Lindner, Mark McCahill, and Daniel Torrey, "The Internet Gopher protocol: a distributed document search and retrieval protocol," University of Minnesota Microcomputer and Workstation Networks Center, Spring 1991.

[2] Tim Howes, Steve Hardcastle-Kille, Wengyik Yeong, & Colin Robbins, "The String Representation of Standard Attribute Syntaxes," Internet Draft, December 1992.

[3] International Organization for Standardization, Information Processing Systems—Open Systems Interconnection—The Directory, International Standard 9594.

[4] International Organization for Standardization, Information Processing Systems—Text Communications—Remote Operations, Part 1: Model, Notation and Service Definition, International Standard 9072-1.

[5] North American Directory Forum, "An X.500 Naming Scheme for National DIT Subtrees and its Application to c=US," Standing Document 5.

[6] Marshall T. Rose, *The Little Black Book: Mail Bonding with OSI Directory Service,* Prentice Hall 1991.

[7] Wengyik Yeong, Tim Howes, & Steve Hardcastle-Kille, "Lightweight Directory Access Protocol," Internet Draft, December 1992.

[8] Benford, S,. "Components of OSI: X.500 Directory Services," *ConneXions,* Volume 3, No. 6, June 1989.

[9] Deutsch, P. & Emtage, A., "The *archie* System: An Internet Electronic Directory Service," *ConneXions,* Volume 6, No. 2, February 1992.

[10] Kahle, Brewster, "An Information System for Corporate Users: Wide Area Information Servers," *ConneXions,* Volume 5, No. 11, November 1991.

[11] Mark McCahill , "The Internet Gopher: A Distributed Server Information System," *ConneXions,* Volume 6, No. 7, July 1992.

TIMOTHY A HOWES holds a B.S.E and M.S.E from the University of Michigan. Since 1989 he has worked for the U-M Information Technology Division on a variety of Unix, TCP/IP, and OSI network programming and protocol design projects. He is currently in charge of X.500 development and deployment on the U-M campus. He is co-chair of the IETF Integrated Directory Services working group, and member of the ACM and IEEE. He can be reached as `tim@umich.edu`.

ITU's TELEDOC Project

by Robert Shaw, International Telecommunication Union

Introduction

This article is an overview of the TELEDOC electronic document exchange project at the *International Telecommunication Union* (ITU). The principle goal of TELEDOC is to provide for remote electronic access to ITU documents.

The ITU is a United Nations agency based in Geneva, Switzerland. It consists organizationally of five permanent organs: the *General Secretariat,* the *International Frequency Registration Board* (IFRB), the *International Radio Consultative Committee* (CCIR), the *International Telegraph and Telephone Consultative Committee* (CCITT) and the *Telecommunications Development Bureau* (BDT). In the beginning of 1993, the ITU is undergoing reorganization into three sectors: development, standardization and radiocommunication. The standards-setting activities of the CCITT and CCIR have been consolidated into a new *Telecommunications Standardization Sector.* The remainder of CCIR activities are being integrated into a new *Radiocommunication Sector* along with the activities of the IFRB. The functions of the new Development Sector are assumed by the BDT. Because the ITU's reorganization was not finalized when this article was written, reference will still be made to the ITU's previous structure (e.g., CCITT, CCIR).

TIES

The ITU's goal is to foster and facilitate the global development of telecommunications through the rule of law, mutual consent and cooperative action. Therefore, supporting information exchange between participants in ITU's work and providing access to telecom-related information is part of its mission. Toward this objective, the ITU started, several years, ago a project called TIES (*Telecom Information Exchange Services*). TIES provides an underlying infrastructure necessary for ITU electronic services including electronic mail, a virtual terminal (VT) interface with file transfer capabilities and an FTP server. The TELEDOC project was initiated in 1992 by ITU's *Information Systems Steering Committee* to meet the pressing requirements for electronic document exchange. TELEDOC is the document exchange service of TIES.

Project requirements

The requirements for TELEDOC can be essentially derived from the requirements of two user groups: those participating in ITU's standardization, radiocommunication and development activities and those interested in ITU publications (e.g., CCITT and CCIR Recommendations).

For example, the first group's requirements might be related to access to information and documents that assist in the development of standards. The document exchange and revision cycle, whether on paper or electronically, has always been the most successful paradigm for standards-making. Since almost all documents now have an electronic form and ITU's standards-makers are geographically dispersed (coming from potentially 175 countries), electronic document exchange is seen as having considerable potential for speeding up standards-making. However, there are considerable challenges due to factors such as weaknesses in networking infrastructures (e.g., especially in developing countries), differences in information technology "cultures," and lack of accepted standards for transparent document conversion.

continued on next page

ITU's TELEDOC Project *(continued)*

For the second group, electronic access to ITU publications such as CCITT and CCIR standards is seen as the most pressing requirement. Besides the considerable technical challenges, this is a political "hot potato." Many people (such as my occasional beer-drinking partners, Carl Malamud and Tony Rutkowski) argue passionately and eloquently that international and national standards should be widely and freely available electronically. Others, such as the governing member bodies that define ITU policy and provide its funding, insist that "appropriate provision of payment" must be made.

The ITU, like many information providing organizations, is in transition from a paper-publishing orientation to a more flexible information dissemination environment. New dissemination possibilities include delivery on magnetic media, CD-ROM or via networks. Although electronic publications delivery over networks (such as the Internet) is an attractive option, there are many issues to consider such as copyright, formats, networking standards and pricing. In the meantime, the ITU is experimenting with new techniques for electronic publications delivery. For example, during 1993 approximately 200 CCITT Recommendations each in the three working languages of the ITU (English, French and Spanish) are being made freely available through the TELEDOC project.

Project elements

When the TELEDOC project was started, the project team decided to break it up into several development components:

- ITU Document Store
- Internal Interface to the ITU Document Store
- External Interfaces to the ITU Document Store

Document Store

TELEDOC required a document reference system to organize ITU documents to be made available for electronic retrieval. The project team envisioned that this reference system would allow distributed users within the ITU to logically group together documents and associate with them attributes (e.g., title, creation date, last modification date, file format, size). This would allow internal and external users of the reference system to locate and retrieve documents by specifying attribute values.

For the purposes of the project, we realized a requirement for a very liberal interpretation of the term "document." In fact, we wanted the possibility to make electronically available any type of ITU "information resource" that could be put into electronic form. This included ASCII and word processing files, database listings, spreadsheets, graphics, ITU publications such as CCITT/CCIR Recommendations, source code or executable programs. We also realized that the required document reference system would need to sometimes support multiple stored formats of the same "conceptual" document. For example, ASCII, *Word* for Windows and *PostScript* versions of the same texts could be made available depending on intended usage (e.g., quick reference, editing or reading). Preferably these multiple stored formats would be linked together to share common attribute information (e.g., document title).

A search was made for commercial products that would meet the needs of the reference system. Complicating factors were mandatory links to an internal document tracking system and the ITU's multilingual environment (English, French and Spanish are working languages).

Unfortunately, within the project's tight time constraints, none were found that had the flexibility or extendibility required. Therefore, the decision was made to implement an in-house solution.

DFR Fortuitously, after the project was started in the beginning of 1992, a new International Standard was published that provided a conceptual model for the planned reference system. This was the International Organization for Standardization's (ISO) *Document Filing and Retrieval Standard* (ISO 10166) more commonly known as DFR.

We were able to quickly obtain a copy of the standard from an ISO contact across the street by trading it for a pizza in the local pizzeria (an example of creative standards pricing and delivery). In fact, following this stream of thought, perhaps pizza delivery should be the paradigm for electronic standards delivery ("Your standard delivered within 30 minutes or you pay nothing!")

Anyway, to quote from the introduction of the DFR standard:

> "The Document Filing and Retrieval (DFR) application provides the capability for large capacity non-volatile document storage to multiple users in a distributed office system...
>
> Documents have associated attributes, to facilitate and control retrieval. Use of these attributes according to given algorithms will enable documents in the document storage to be browsed, retrieved, managed and deleted in a variety of ways. Access control protects documents from unauthorized operations. Documents can be stored in nested groups. With specific attributes a document can be designated a version of another document. Single documents, references or groups can be moved from one group into another group. Enumeration of groups, identification by other attributes besides names, identification by conditions over attributes, search for documents meeting search criteria, concurrent access to the same documents, reference or group of documents are further functions provided by this standard for the user requirements in an office environment."

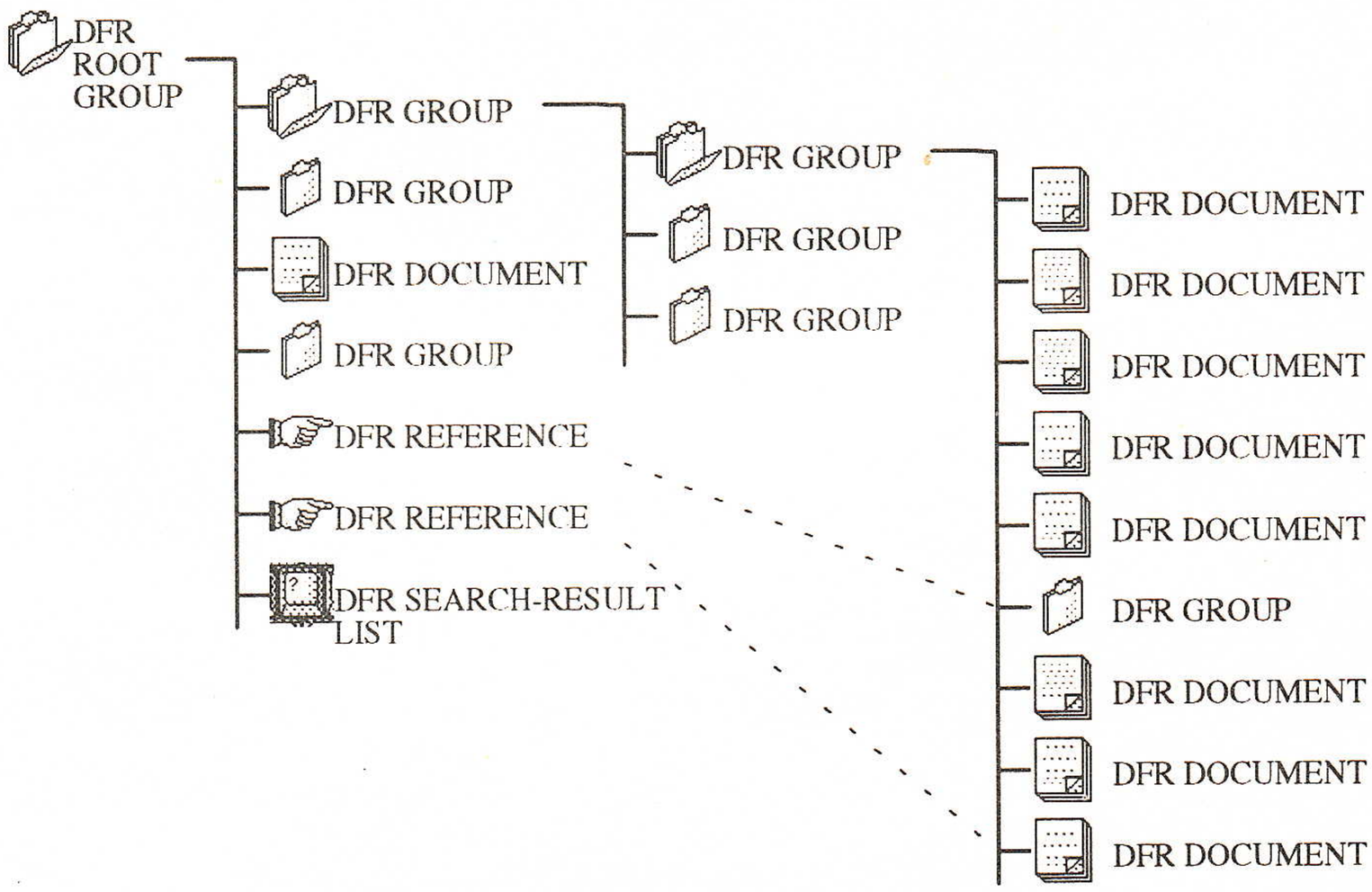

Figure 1: DFR Document Store

continued on next page

ITU's TELEDOC Project *(continued)*

Structure

DFR uses the terminology *Document Store* to refer to a collection of objects logically arranged in a hierarchical structure. These objects include:

- *DOCUMENT:* For example, an ASCII, *PostScript* or *Word* for Windows file;
- *GROUP:* For example, group of CCITT documents or group of CCIR documents;
- *REFERENCE:* For example, a reference to a document in another group;
- *SEARCH-RESULT-LIST:* For example, a list of documents meeting a search-query on attributes.

A pictorial representation of a hierarchical DFR Document Store with these object types is shown in Figure 1. Note all objects are subordinate to the DFR-Root-Group.

The project team envisioned an ITU Document Store matching the DFR model. The ITU Root Group would be divided into groups representing the main organizational structure of the ITU. For example, one group would be named "CCITT" and this group would contain CCITT-related groups and documents. Under this group, there would be other groups including CCITT Recommendations and CCITT Study Groups. A partial pictorial representation of this envisioned ITU Document Store is illustrated in Figure 2.

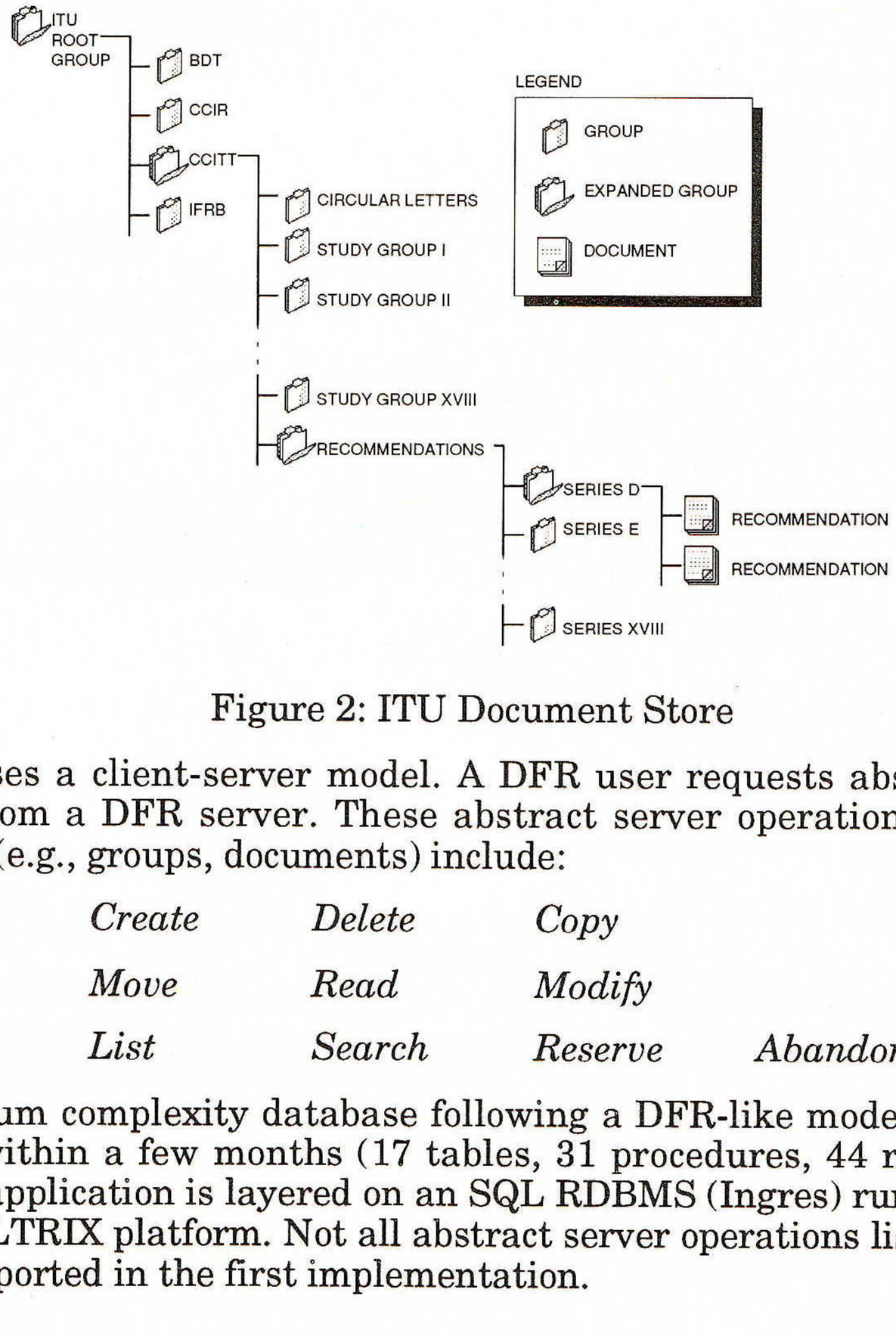

Figure 2: ITU Document Store

DFR uses a client-server model. A DFR user requests abstract services from a DFR server. These abstract server operations on DFR objects (e.g., groups, documents) include:

Create	*Delete*	*Copy*	
Move	*Read*	*Modify*	
List	*Search*	*Reserve*	*Abandon*

A medium complexity database following a DFR-like model was designed within a few months (17 tables, 31 procedures, 44 rules). The server application is layered on an SQL RDBMS (Ingres) running on a DEC ULTRIX platform. Not all abstract server operations listed above are supported in the first implementation.

Internal interface

Fundamental to the success of keeping the ITU Document Store up-to-date would be a good internal client interface. Most of the 700 ITU staff members have a 386/486 PC on their desk running MS-Windows 3.1. They are connected to the ITU's DECnet-based LAN with VAX/VMS and ULTRIX servers. ITU staff use a suite of Windows-based office automation tools such as Microsoft *Word* for Windows and *Excel*. *Word* for Windows is used both for normal administrative word processing tasks and for the final preparation of most ITU publications (e.g., CCITT Recommendations). We decided it would be ideal if we could integrate the internal interface directly within ITU's word processing package, *Word* for Windows. This was successfully done using *Word*'s macro language and extensions in MS-Windows dynamic link libraries (DLLs). The support for database SQL APIs over the ITU's DECnet network to the ULTRIX server is via a Windows DLL interface called *SequeLink*. This product supports a wide variety of clients/network protocols/servers/RDBMS engines. Shown below is the main graphical interface of the client we built.

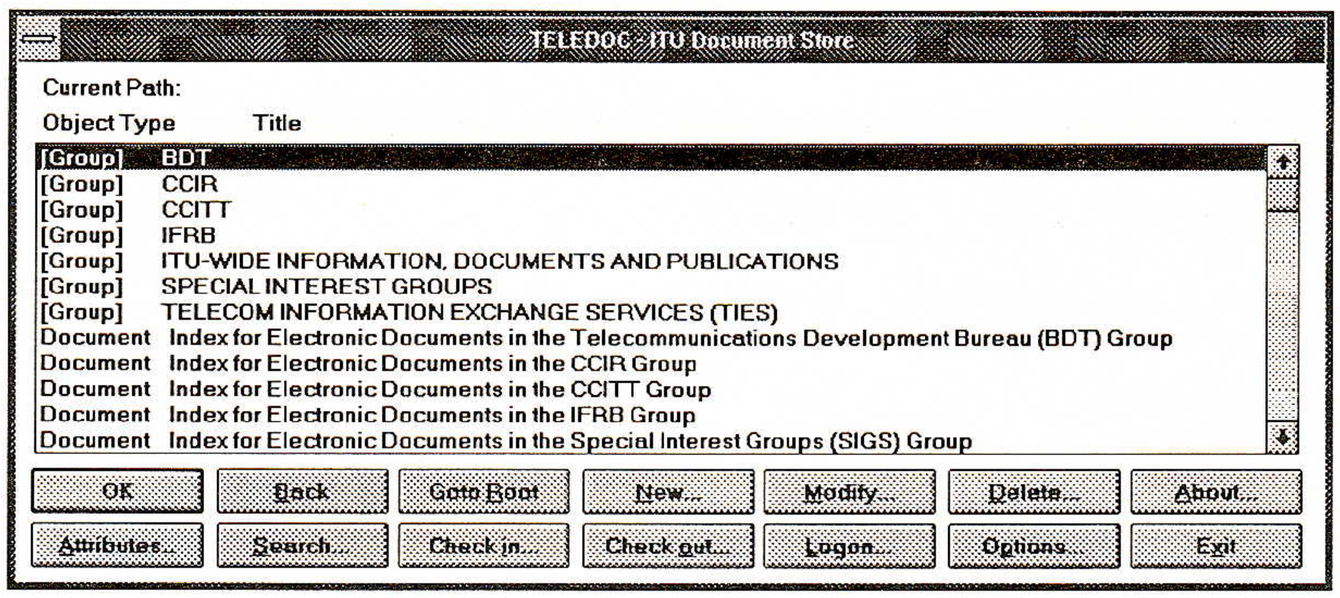

Figure 3: TELEDOC Internal Client: Main Menu

Only authorized users can create, delete or update groups or documents. Security is currently through SQL grant privileges on the database but more sophisticated access control is under development. The internal client permits any ITU user with *Word* for Windows to browse up and down the hierarchical tree of groups and documents. When groups are created, they are assigned a long title and a UNIX-like directory path (the path only exists as an "attribute"). This path is used as shorthand notation to refer to any location (it is also used to generate the external FTP server interface). For example, */ccitt/rec/x* refers to the CCITT Recommendations X Series group. Shown below is the internal interface after a user has selected this group. Note the displayed path at the top.

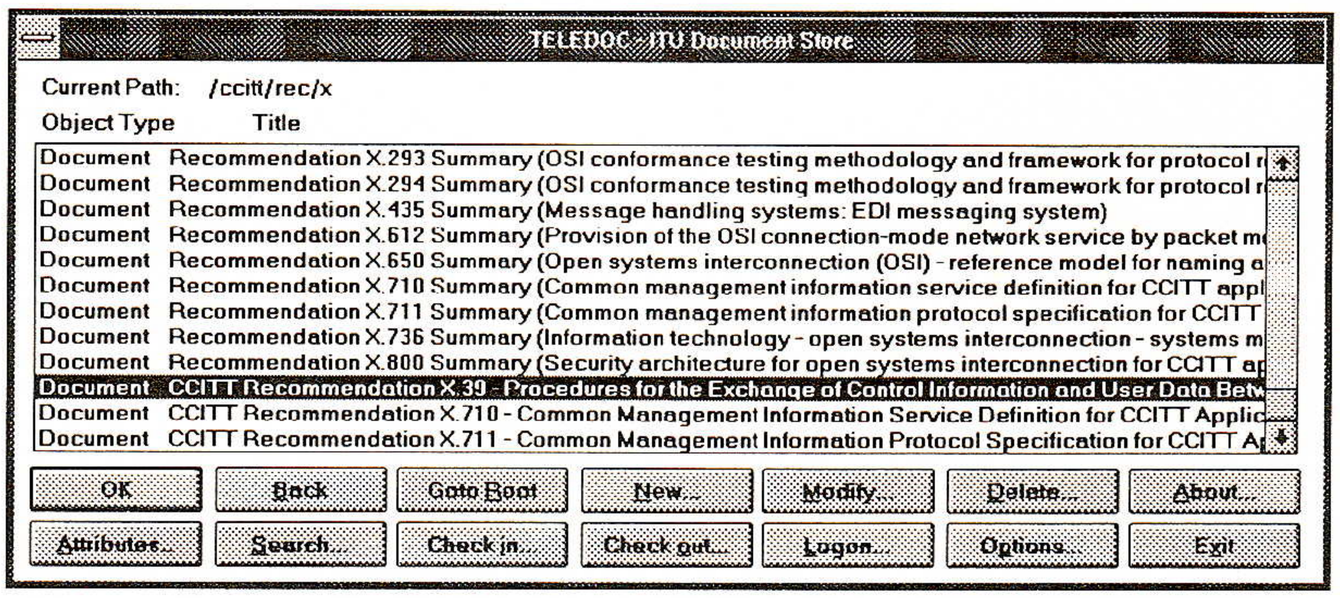

Figure 4: TELEDOC Internal Client: Group Selected

continued on next page

ITU's TELEDOC Project *(continued)*

The attributes can be also queried for any object. For example, the attributes for the actual CCITT Recommendation document highlighted above are shown below.

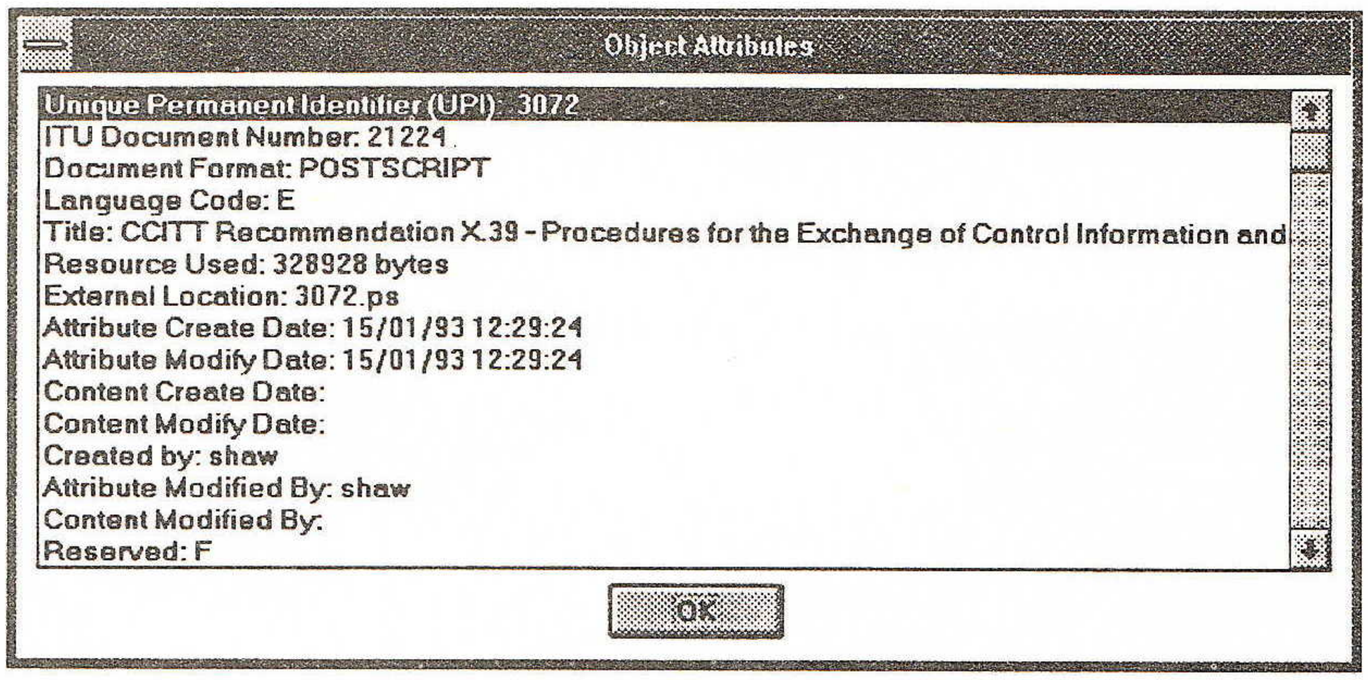

Figure 5: TELEDOC Internal Client: Document Attribute

The *Unique Permanent Identifier* (UPI) for each stored document is generated by the server component when the document is added. It is used as the unique retrieval code for each stored document (e.g., using the electronic mail interface described below). Multiple stored format versions of the same document are linked together through the attribute ITU Document Number. The same stored document can be put into several groups supporting different user "search knowledge paths" to the same information.

Any "document" on the Store can be opened with the appropriate tool if present on the internal user's PC. For example, double-clicking with a mouse on a *Word* for Windows document causes a copy of the document to be moved from the ITU Document Store to the local PC and opened in *Word*. Likewise, double-clicking on a graphics file loads the corresponding graphics editor. If the user does not have the right tool to manipulate or visualize an object, it is optionally copied to the user's PC. For example, a *PostScript* file can be copied from the Store to the user's PC and then later copied to a *PostScript* printer.

External interfaces

TELEDOC external interfaces determine how remote users can interact with the ITU Document Store. Several TELEDOC external interfaces are being implemented including:

- Electronic mail
- Internet FTP
- Interactive VT interface

TAM

The TELEDOC *Auto-Answering Mailbox* (TAM) is the electronic mail interface. It is a "robot" mailbox running 24 hours a day at ITU headquarters in Geneva. It is an application layered on both an X.400 mail user agent and a special client to the ITU Document Store. The environment is Windows 3.1 on a dedicated Compaq 486/50 with 16M bytes of memory. The application is written in *Word* for Windows macro language and uses DEC's *MailWorks* X.400 user agent. *MailWorks* is a client–server application with an X.400 MTA (Message Transfer Agent) located on a VAX running VMS. *Word* and *MailWorks* communicate via MS-Windows *Dynamic Data Exchange* (DDE). *Word* handles the dynamic formatting of reply messages and certain kinds of message attachments (e.g., the results of a query for a list of documents available in a group).

MailWorks provides the APIs for receiving and sending mail. Internet RFC 822 addressed mail is currently gatewayed through the SWITCH gateway (SWITCH is the Swiss university network). The X.400 address of the TAM is:

```
S=teledoc; P=itu; A=arcom; C=ch
```

The TAM can also be addressed using an RFC 822 address:

```
teledoc@itu.arcom.ch
```

Electronic mail to the TAM should only contain simple commands to interface with the ITU Document Store. When mail arrives, the TAM examines who the mail came from, accordingly processes commands in the message body, then mails back responses.

For example, it is possible to request a list (LIST) of groups and documents in any group of the ITU Document Store. The argument for the LIST command is the UNIX-like pathname constructed on group creation. The TAM constructs a list of currently available documents with information such as title, format, size, and the UPI code used for retrieval. Only documents that have an attribute as being "public" are shown. This list is e-mailed back to the requester. After the user reviews the list of available documents, the desired document can be retrieved with the GET command and the corresponding UPI code. Here is an example message to the TAM:

```
-----------------------------------------------
To: teledoc@itu.arcom.ch
FROM: (ME)
SUBJECT: (IGNORED)
-----------------------------------------------
START
HELP
LIST
LIST CCITT
LIST CCITT/REC/D
GET 1449
END
-----------------------------------------------
```

The above message to the TAM asks to:

- Send a help file explaining TAM commands
- Send a list of groups and/or documents at the root level of the ITU Document Store
- Send a list of groups and/or documents in the CCITT group
- Send a list of groups and/or documents in the CCITT Recommendations D Series group
- Send a document with a UPI of 1449

Up to 50 lines per message are processed. Any message sent to the TAM which does not contain at least one valid command will cause the help file to be sent back.

Index files giving a "snapshot" view of the ITU major groups (e.g., CCITT, CCIR) hierarchy and the documents available are available (currently in the root of the ITU Document Store).

continued on next page

ITU's TELEDOC Project *(continued)*

When a document is requested for retrieval with the GET command, the Document Store is queried as to the document type (format) and a lookup defines how to tag it when sent out as an X.400 message body part (e.g., ASCII (IA5) or Binary). Non-ASCII files returned to Internet e-mail addresses are automatically UUENCODED to an ASCII form "on the fly" by the TAM. Eventual support for the Internet MIME standard is planned. Many X.400 correspondents directly retrieve binary files as undefined binary message body parts (X.400 Body Part 14). However, costs are proving to be very high for sending out large documents and there may be steps to encourage the use of other external interfaces once these are available (i.e., interactive VT or FTP).

Planned enhancements for the TAM include "on the fly" file compression with user-specified UNIX, MS-DOS or Mac "flavors." There are also plans to implement a user registration facility which will permit simple restricted access to certain groups and/or documents by matching incoming address attributes against previously registered values. The user registration facility is planned to be valid across the three external interfaces planned: electronic mail, VT, and FTP.

VT interface

An interactive interface to the ITU Document Store is now under development. This will become available under the Telecom Information Exchange Services (TIES) VT interface. The interactive interface will allow browsing through groups and documents and reading ASCII-formatted documents on-line. It will be possible to download non-viewable formats (e.g., binary files) using *Kermit* file transfer.

The TIES VT interface can be accessed over dial-up, X.25, and *telnet* connections. In addition to TELEDOC, there are other services available (some only by subscription). TIES users must be registered before having access. There is currently no charge for general access. Information on TIES registration can be obtained from:

Information Services Department Secretariat
ITU
Place des Nations
1211 Geneva 20
Switzerland
FAX: +41 22 730 5337
X.400: `S=helpdesk; A=arcom; P=itu; C=ch`
Internet: `helpdesk@itu.arcom.ch`

The ITU has installed its FTP server and is now building the interface from the ITU Document Store. The current plan is for a copy of the ITU Document Store to be made on a regular basis to the FTP server. The short hand path notation used to refer to Document Store groups will be used to create and maintain the server directory structure and to associate "documents" with directories. Index files with comprehensive attribute information (e.g., group, document titles) will describe the structure and contents of the server. Users are required to have a TIES user account to use the FTP server interface. To register, use the address mentioned above.

Formats

For TELEDOC, we had to think about what formats (or as document processing aficionados like to say, "architectures") we would support. Unfortunately, there is no such thing as "universal" document exchange format. ASCII, a *de facto* standard, is OK for the exchange of simple text but is totally inappropriate for documents containing multiple fonts, complex formatting, multilingual text or such objects as equations, tables and graphics (e.g., as in many CCITT and CCIR standards).

ROBERT SHAW is the TELEDOC Project Coordinator at the ITU in Geneva, Switzerland. He lives just across the border in France. He attended the University of Southern California and the University of Dijon, France. His favorite relaxation is sitting with a glass of wine (preferably a few year old Volnay) and listening to classical music while sleuthing the history of 19th and 20th century landscape and genre paintings. E-mail:
822: shaw@itu.arcom.ch
X.400: G=Robert; S=shaw; A=arcom;P=itu; C=ch

The very healthy word processing marketplace with millions of units shipped per year has essentially created a "Tower of Babel" situation. When you add up multiple platforms, multiple formats and multiple versions, you get multiple problems. This is not a reflection on the inadequacies of document conversion software: it is a fundamental problem of document architecture. It is impossible to convert from a rich architecture to a poorer one (or vice versa) and expect transparency. Sometimes, if you get anything reasonable it's surprising.

Although not quite as complex, the same problems exist for electronic delivery of publication quality documents (the difference being that these are typically non-revisable). Here there is a tremendous marketplace potential which software suppliers are starting to recognize. For example, Adobe now appears trying to leverage its dominance of the *page description language* (PDL) market with new cross-platform technology called Acrobat. The goal of Acrobat is to deliver publication quality documents on UNIX, Windows and Macintosh platforms through the use of "viewers." On a smaller scale, many companies are now giving away proprietary document viewers in order to escape the "ASCII jail" and deliver quality documents over networks (often with compression and hypertext features thrown in for free). For proof, take a look at what is happening on *CompuServe*'s vendor forums.

ODA and SGML

In the international standards world, there are essentially two document standards: *Standard Generalized Markup Language* (SGML)—ISO 8879 and *Open Document Architecture* (ODA)—ISO 8613/CCITT T.410 Series. SGML is not a document architecture but rather a "language" to describe any document architecture. Since there are many SGML architectures (called SGML *applications*), it is not a solution for blind document interchange. ODA has the potential of solving the problems of blind interchange but it is a very complex standard with currently very few implementations. Recognizing this, the *Open Document Architecture Consortium* (ODAC) formed by major companies like IBM, DEC, ICL, and BULL have announced the availability of an openly licensed toolkit for manipulation of ODA documents. Products based on this toolkit probably won't become available until late 1993 or 1994 and the market acceptance remains to be seen.

Until an industry or international standard format dominates the electronic document delivery market (as *PostScript* dominates the current PDL market), it will be necessary to support multiple formats for electronic document delivery. For TELEDOC, different formats are posted according to the class of document and intended usage. For example, if a document can be represented in ASCII, it is posted that way for ease of access and use. If document revisability is required, the formatting requirements are complex or it must be read on multiple platforms, Microsoft *Rich Text Format* (RTF) is used. If publication quality formatting is required and fidelity to a corresponding paper publication, it is posted in *PostScript* format.

References

[1] Malamud, C., *Exploring the Internet—A Technical Travelogue,* Prentice-Hall, 1992, ISBN 0-13-296898-3.

[2] Malamud, C. "The ITU Adopts a New Meta-Standard: Open Access," *ConneXions,* Volume 5, No. 12, December 1991.

[3] Stewart, J. "Components of OSI: Document Interchange Using ODA," *ConneXions,* Volume 5, No. 8, August 1991.

[4] Stewart, J. & Bramhall, M., "Comparing Compound Document Processing Models," *ConneXions,* Volume 6, No. 8, August 1992.

[5] Rutkowski, A., "Networking the Telecom Standards Bodies," *ConneXions,* Volume 5, No. 9, September 1991.

IGOSS/GOSIP Comments Solicited

Introduction

The *National Institute of Standards and Technology* (NIST) has announced the availability of the *Industry Government Open Systems Specification* (IGOSS) for public review and comment. Industry and government organizations including NIST, the Canadian Government, the World Federation of MAP/TOP User Groups, and the electric power industry have been working together to develop a common open systems specification for the acquisition of computer networking products and services based on the international *Open Systems Interconnection* (OSI) standards.

Profiles

In the past, different OSI profiles have been developed for the U.S. Government, the Canadian Government, and for industry organizations. NIST developed the U.S. *Government OSI Profile* (GOSIP), which has been issued as U.S. Federal Information Processing Standard 146–1, for use by the Federal Government in its acquisition of computer networking products and services. The Treasury Board Secretariat of Canada published the *Canadian Open Systems Application Criteria* (COSAC). The MAP and TOP User Groups published their OSI specifications, and the Electric Power Research Institute published the *Utility Communications Architecture* (UCA) document.

The proposed Industry Government Open Systems Specification will provide common requirements for OSI products and services, and enable the computer industry to develop products that meet the requirements of a broad user market.

The organizations participating in the development of IGOSS expect to base their future OSI requirements documents on the IGOSS, specifying individually how IGOSS will be used within their communities. The IGOSS is expected to become the primary reference for functional profiles to be issued by the individual IGOSS organizations in the future.

Comment period

The participants in the development of IGOSS are soliciting comments from interested parties on the technical content of the IGOSS. Written comments should be sent prior to May 15, 1993 to: Mr. Gerard F. Mulvenna, National Institute of Standards and Technology, Technology Building, Room B217, Gaithersburg, MD 20899, telephone 301-975-3631. Comments may also be submitted electronically to the Internet mailbox:

```
igoss-comments@osi.ncsl.nist.gov
```

Documents

Copies of the IGOSS may be obtained from the Standards Processing Coordinator (ADP), National Institute of Standards and Technology, Technology Building, Room B-64, Gaithersburg, MD 20899, telephone 301-975-2816.

IGOSS is also available on-line. To access via anonymous FTP, address is `osi.ncsl.nist.gov` (129.6.48.100). IGOSS is located in directory `/pub/igoss`. The file names are `igoss_v1.asc` (ASCII file), `igoss_v1.ps` (*PostScript* file) and `igoss_v1.ps.Z` (compressed *PostScript* file). You must retrieve one of the *PostScript* files in order to obtain all figures.

To access via anonymous FTAM use:

```
Paddr =
{1,1,1,47:0005:80:005a00:0000:0001:e137:080020079efc:00}
userid = anon,
realstore = unix
```

If using ISODE, the corresponding "isoentities" entry is:

```
osi.ncsl.nist.govfilestore NULL \
#1/#1/#1/NS+47000580005a0000000001e137080020079efc00
```

FTAM using RFC 1006 is also supported on `osi.ncsl.nist.gov`.

GOSIP

The U.S. GOSIP is expected to be issued for public comment in the fourth quarter of FY 1993. Because GOSIP will be promulgated as a *Federal Information Processing Standard* (FIPS), there will be a Federal Register notice announcing the public comment period. The intent is that all comments on IGOSS be made during the IGOSS public comment period and that comments during the GOSIP public comment period be limited to the additional material in the GOSIP. GOSIP will reference IGOSS to specify the common OSI procurement requirements for the IGOSS organizations; however, GOSIP will also contain additional specifications required by the U.S. Federal Government but not agreed to in common. This additional information is expected to be limited to security and registration issues. GOSIP will also contain the Federal Government applicability statement which will continue to be a mandate.

Following the successful resolution of comments pertaining to both documents, GOSIP will be promulgated as a FIPS and IGOSS will be published as a NIST Special Publication sometime in FY 1994.

"Components of OSI" in *ConneXions*

The following articles have appeared in *ConneXions* under the heading "Components of OSI":

Integrated Services Digital Network (ISDN)	April	1989
X.400 Message Handling System	May	1989
X.500 Directory Services	June	1989
The Transport Layer	July	1989
Routing overview	August	1989
IS–IS Intra-Domain Routing	August	1989
ES–IS Routing	August	1989
The Session Service	September	1989
Connectionless Network Protocol (CLNP)	October	1989
The Presentation Layer	November	1989
A taxonomy of the players	December	1989
The Application Layer Structure	January	1990
File Transfer, Access, and Management (FTAM)	April	1990
The Security Architecture	August	1990
Group Communication	September	1990
X.25—the Network, Data Link, & Physical Layers	December	1990
The Virtual Terminal ASE	January	1991
Systems Management	April	1991
CO/CL Interworking	May	1991
Open/Office Document Architecture (ODA)	August	1991
Abstract Syntax Notation One (ASN.1)	January	1992
Broadband ISDN	April	1992
Synchronous Optical Network (SONET)	April	1992
Asynchronous Transfer Mode (ATM)	April	1992
Inter-Domain Routing Protocol (IDRP)	May	1992
The Remote Procedure Call (RPC) Service	June	1992
OSI Conformance Testing	December	1992
International Standardized Profiles	January	1993

Announcement and Call for Papers

The Fourth *USENIX UNIX Security Symposium* will be held October 4–7, 1993 at the Santa Clara Marriott in Santa Clara, California. The Symposium is hosted by USENIX in cooperation with the *Computer Emergency Response Team* (CERT) Coordination Center.

Goal

The goal of this symposium is to bring together security practitioners, system administrators and system programmers, and anyone with an interest in computer security as it relates to networks and the UNIX operating system. The symposium will consist of tutorials, invited speakers, technical presentations, and panel sessions.

Format

This will be a three and 1/2 day, single-track symposium. The first day will be devoted to tutorial presentations. The following two and 1/2 days will include technical presentations and panel sessions. There will also be two evenings available for birds-of-a-feather sessions and work-in-progress sessions.

Topics

Papers are being solicited in areas including but not limited to:

- User/system authentication
- File system security
- Network security
- Security and system management
- Security-enhanced versions of the UNIX operating system
- Security tools
- Network intrusions (including case studies and intrusion detection efforts)
- Security on high-bandwidth networks

Important dates

Extended abstracts due:	June 4, 1993
Program Committee decisions made:	June 30, 1993
Camera-ready papers due:	August 15, 1993

Submissions

Send hardcopy submissions to the program chair:

Bill Cheswick
AT&T Bell Laboratories
Room 2c416
600 Mountain Avenue
Murray Hill, NJ 07974
USA

Send ASCII or *PostScript* submissions to: `ches@research.att.com`

Program Committee

Steve Bellovin	AT&T Bell Laboratories
Matt Bishop	Dartmouth College
Ed DeHart	CERT
Jim Ellis	CERT
Marcus Ranum	Trusted Information Systems

Call for Papers

Focus

Matt Bishop will be Guest Editor of a special issue of the journal *Computing Systems* to be published in 1993. The issue will be devoted to "Security and Integrity of Open Systems." Papers on all aspects of policy, issues, theory, design, implementation, and experiences with security and integrity in open systems are solicited for the issue. The deadline for submissions is June 1, 1993; papers submitted after this deadline will not be considered. Prospective authors should send five copies of their papers to:

Professor Matt Bishop
Mathematics and Computer Science
Dartmouth College
6188 Bradley Hall
Hanover, NH 03755-3551
Phone: +1-603-646-3267
E-mail: Matt.Bishop@dartmouth.edu

Submissions

Submissions should not have appeared in other archival publications prior to their submission. Papers developed from earlier conference, symposia and workshop presentations are welcome.

About the journal

Computing Systems is a journal dedicated to the analysis and understanding of the theory, design, art, engineering and implementation of advanced computing systems, with an emphasis on systems inspired or influenced by the UNIX tradition. The journal's content includes coverage of topics in operating systems, architecture, networking, interfaces, programming languages, and sophisticated applications.

Computing Systems (ISSN 0895-6340) is a refereed, quarterly journal published by the University of California Press for the USENIX Association. USENIX is a professional and technical association of individuals and institutions concerned with breeding innovation in the UNIX tradition.

Now in its fifth year of publication, *Computing Systems* is regularly distributed to 4900 individual subscribers and over 600 institutional subscribers (libraries, research labs, etc.) around the world. Some special-topic issues are often distributed more widely.

The editor-in-chief of *Computing Systems* is Mike O'Dell of Bellcore. Gene Spafford of Purdue University is Associate Editor, and Peter Salus of the Sun User Group is the Managing Editor.

Write to *ConneXions*!

Have a question about your subscription? Are you moving, and need to give us your new address? Suggestions for topics? Want to write an article? A letter to the Editor? Have a question for an author? Need a *ConneXions* binder? Want to enquire about back issues? (there are now more than 70 to choose from; ask for our free 1987–1992 index booklet). We want to hear from you. Contact us at:

ConneXions—The Interoperability Report
480 San Antonio Road, Suite 100
Mountain View, CA 94040–1219
USA
Phone: +1 415-941-3399 or 1-800-INTEROP (Toll-free in the USA)
Fax: +1 415-949-1779
E-mail: connexions@interop.com

CONNEXIONS
480 San Antonio Road
Suite 100
Mountain View, CA 94040
415-941-3399
FAX: 415-949-1779

FIRST CLASS MAIL
U.S. POSTAGE
PAID
SAN JOSE, CA
PERMIT NO. 1

ADDRESS CORRECTION
REQUESTED

CONNEXIONS

EDITOR and PUBLISHER Ole J. Jacobsen